جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Optimization Methods and Software for Federated Learning

PhD Defense

Konstantin Burlachenko
Computer Science, KAUST

Dissertation Defense Committee Members

Peter Richtárik

Eric Feron

David Keyes

Suhaib Fahmy

جامعة الملك عبدالله
للعلوم والتقنية

King Abdullah University of
Science and Technology

Stephen Boyd

Stanford
University

Nic Lane

UNIVERSITY OF
CAMBRIDGE

Images: Google Search

Background

MS in Computer Science

Bauman Moscow State Technical University (2003 — 2009)

Industry Experience

Startup (2012) Acronis (2010 — 2012) Yandex (2013 — 2014)

NVIDIA (2014 — 2019) HUAWEI (2019 — 2020)

Stanford Graduate Certificates

Data, Models and Optimization Graduate Certificate (2015 — 2018)

Artificial Intelligence Graduate Certificate (2016 — 2019)

PhD Academic Journey

Joined Prof. P. Richtárik's Optimization and ML Lab at KAUST (August 2020)

Defended CS PhD Proposal (2022)

Member of Center of Excellence SDAIA-KAUST AI (2022 — 2023)

Internships

Research Scientist Internship Offer, Facebook Inc., Menlo Park, USA (2021)

Internship in Private Federated Learning ML Team, Apple, Cambridge, UK (2024)

Conference Presentations

Presentations: ICLR'24, SIAM'23, ICML'21, NSF-TRIPODS'21, DistributedML'21&'23

Awards

Dean's Award (2020) Grant from SDAIA (2022) Dean's Award (2023)

AMD MI50 from AMD (2023) Shaheen III Proposal (2024) RDIA grant (2025)

Images: Google Search

Papers Co-Authored During PhD (Creation Timeline)

[1]	**Personalized Federated Learning with Communication Compression** E. Bergou, **K. Burlachenko**, A. Dutta, P. Richtárik	Ch6		Transactions on ML Research (tmlr)
[2]	**MARINA: Faster Non-Convex Distributed Learning with Compression** E. Gorbunov, **K. Burlachenko**, Z. Li, P. Richtárik		Symposium on ACM PODC 2022	ICML International Conference On Machine Learning
[3]	**Fl_PyTorch: Optimization Research Simulator for Federated Learning** **K. Burlachenko**, S. Horváth, P. Richtárik	Ch2	Symposium on SIAM OP23	acm; DISTRIBUTEDML
[4]	**Faster Rates for Compressed Federated Learning with Client-Variance Reduction** H. Zhao, **K. Burlachenko**, Z. Li, P. Richtárik			SIAM Journal on Mathematics of Data Science
[5]	**Don't Compress Gradients in Random Reshuffling: Compress Gradient Differences** A. Sadiev, G. Malinovsky, E. Gorbunov, I. Sokolov, A. Khaled, **K. Burlachenko**, P. Richtárik		Workshop FL-ICML-2023	Neural Information Processing Systems
[6]	**Sharper Rates and Flexible Framework for Nonconvex SGD with Client and Data Sampling** A. Tyurin, L. Sun, **K. Burlachenko**, P. Richtárik	Ch5		Transactions on ML Research (tmlr)
[7]	**Federated Learning with Regularized Client Participation** G. Malinovsky, S. Horváth, **K. Burlachenko**, P. Richtárik		Workshop FL-ICML-2023	arXiv; UNDER REVIEW
[8]	**Error Feedback Shines when Features are Rare** P. Richtárik, E. Gasanov, **K. Burlachenko**			arXiv; Preparing for Resubmission
[9]	**Federated Learning is Better with Non-Homomorphic Encryption** **K. Burlachenko**, A. Alrowithi, F. Ali Albalawi, P. Richtárik	Ch4		acm; DISTRIBUTEDML
[10]	**Error Feedback Reloaded: From Quadratic to Arithmetic Mean of Smoothness Constants** P. Richtárik, E. Gasanov, **K. Burlachenko**	Ch3	ML Summer School Okinawa 2024	ICLR International Conference On Learning Representations
[11]	**Unlocking FedNL: Self-Contained Compute-Optimized Implementation** **K. Burlachenko**, P. Richtárik	Ch7	KAUST AI Symposium 2024	arXiv; UNDER REVIEW
[12]	**PV-Tuning: Beyond Straight-Through Estimation for Extreme LLM Compression** V. Malinovskii, D. Mazur, I. Ilin, D. Kuznedelev, **K. Burlachenko**, K. Yi, D. Alistarh, P. Richtárik			Neural Information Processing Systems
[13]	**BurTorch: Revisiting Training from First Principles by Coupling Autodiff, Math Optimization, and Systems** **K. Burlachenko**, P. Richtárik	Ch8		arXiv; UNDER REVIEW

Machine Learning

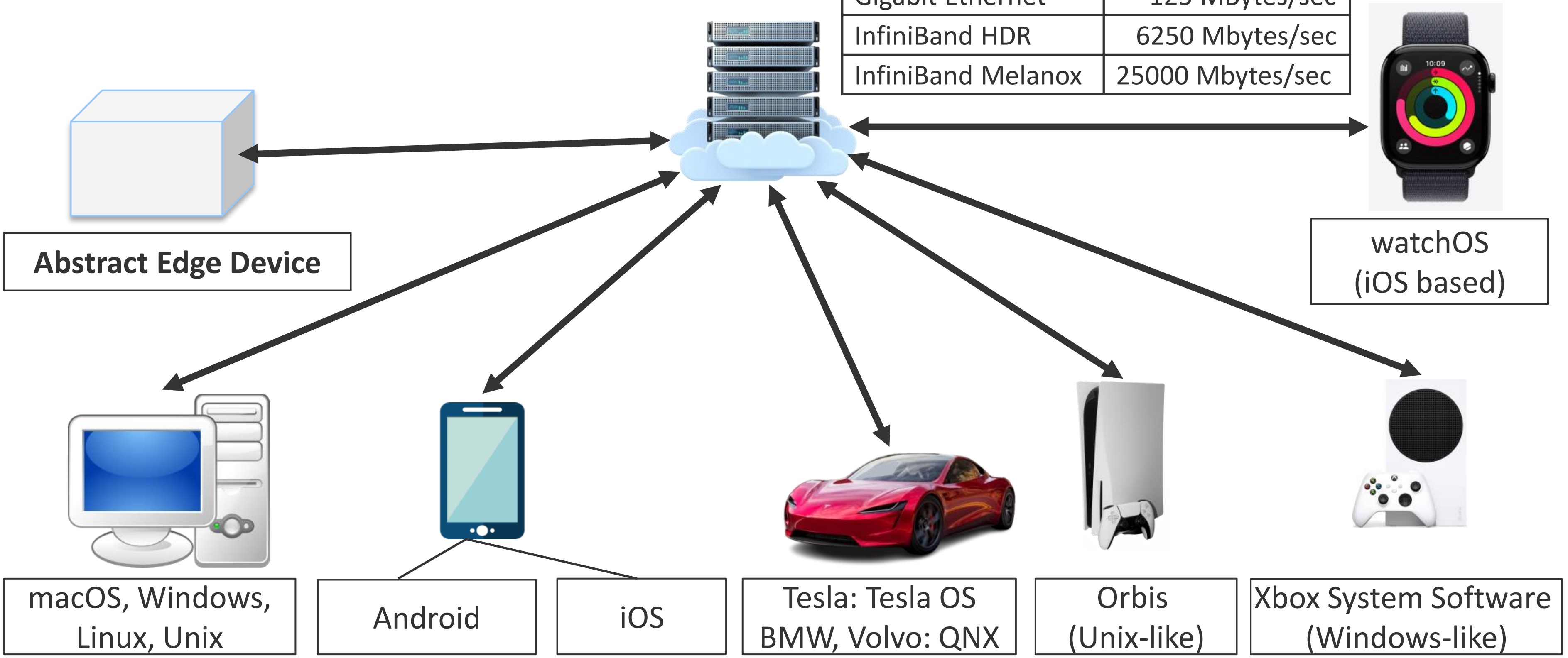
Traditional machine learning assumes that
the training dataset is collected and stored centrally

Traditional machine learning assumes that
the training dataset is collected and stored centrally

However, centralized storage is **not** where data is generated in the first place

Shifting Training to Edge Devices

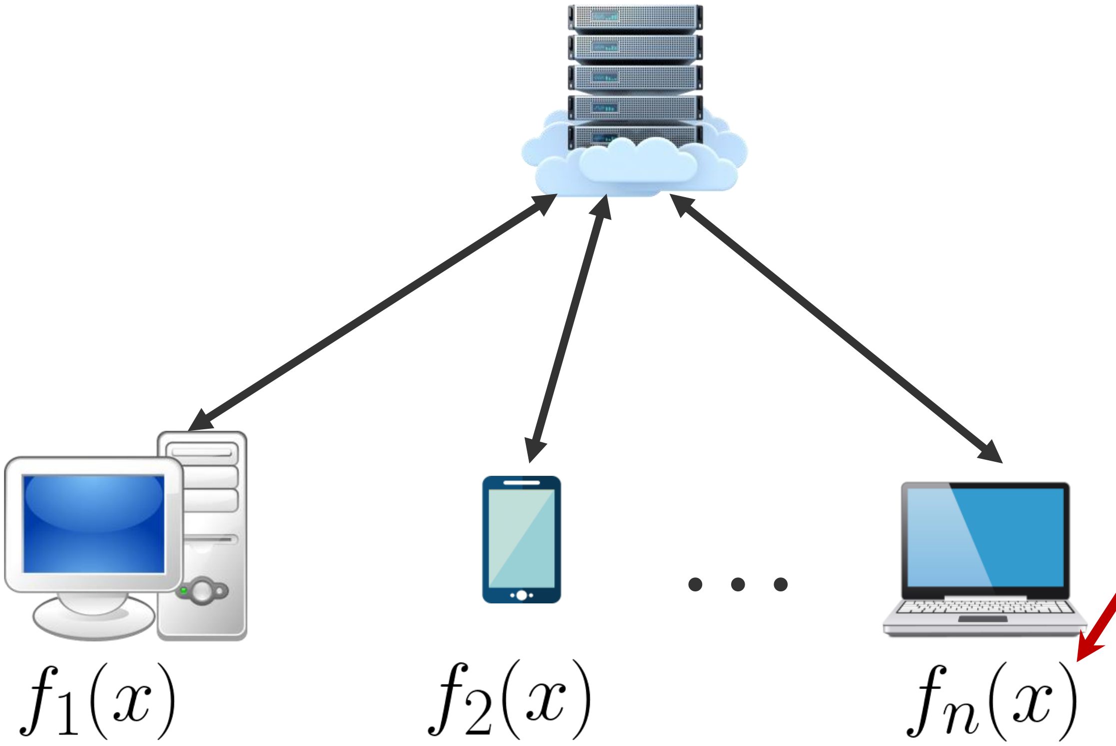
Fast Ethernet	12 MBytes/sec
Gigabit Ethernet	125 MBytes/sec
InfiniBand HDR	6250 Mbytes/sec
InfiniBand Melanox	25000 Mbytes/sec



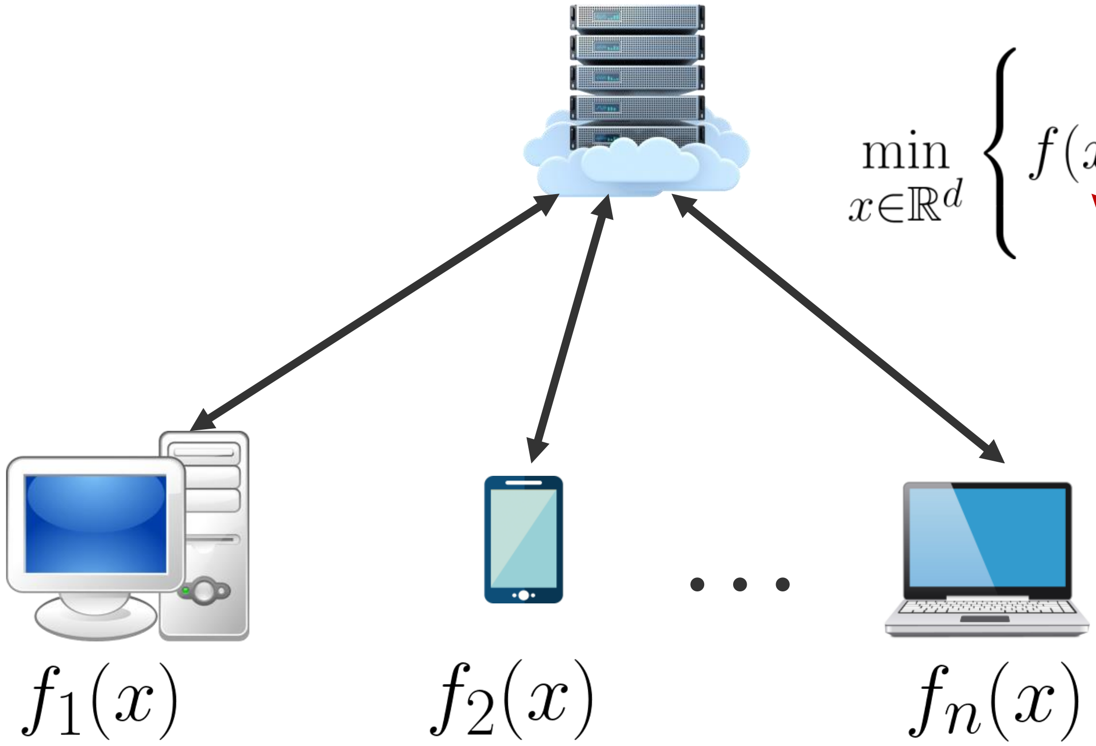
Images: Google Search

Shifting Training to Edge Devices

f_n is a local loss constructed from data D_n



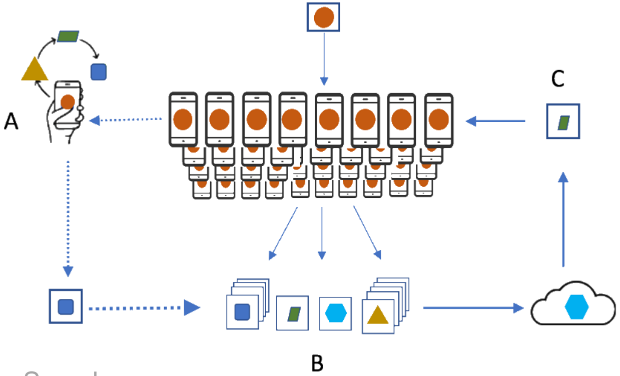
Shifting Training to Edge Devices



$$\min_{x\in\mathbb{R}^d} \left\{ f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}$$

Optimization objective

Federated Learning (FL)



$$\min_{x\in\mathbb{R}^d}\left\{f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\right\}$$

Images: Google Search

FL Origins

Federated Learning: Strategies for Improving Communication Efficiency (**2016**) J. Konečný, B. McMahan, F. X. Yu, P. Richtárik, A.T. Suresh, D. Bacon

Federated Optimization: Distributed Machine Learning for On-Device Intelligence (**2016**) J.Konečný, B. McMahan, D. Ramage, P. Richtárik

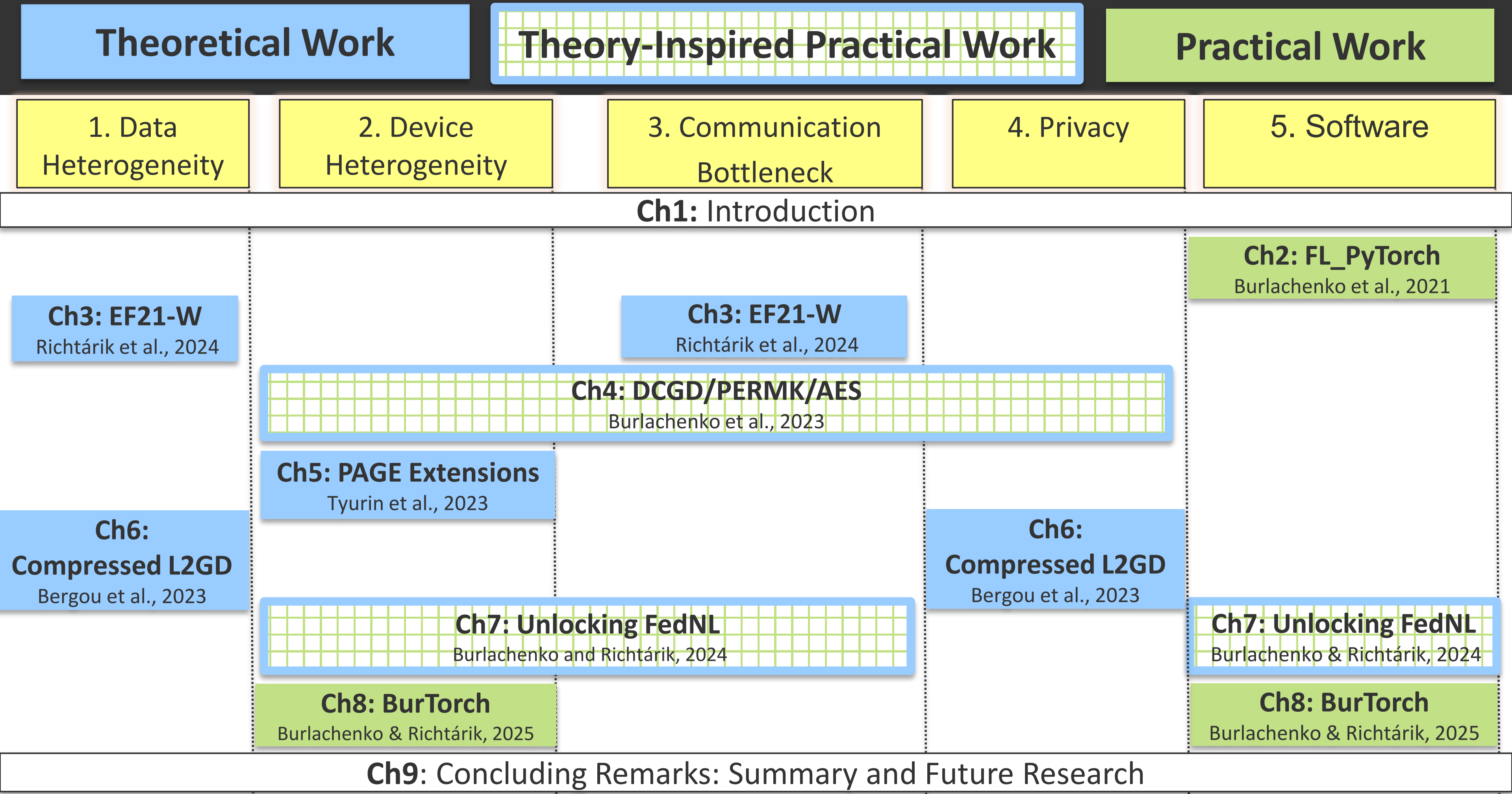
Communication-Efficient Learning of Deep Networks from Decentralized Data (**2017**) B.McMahan, et al.

Advances and Open Problems in Federated Learning (**2021**) P. Kairouz, et al.

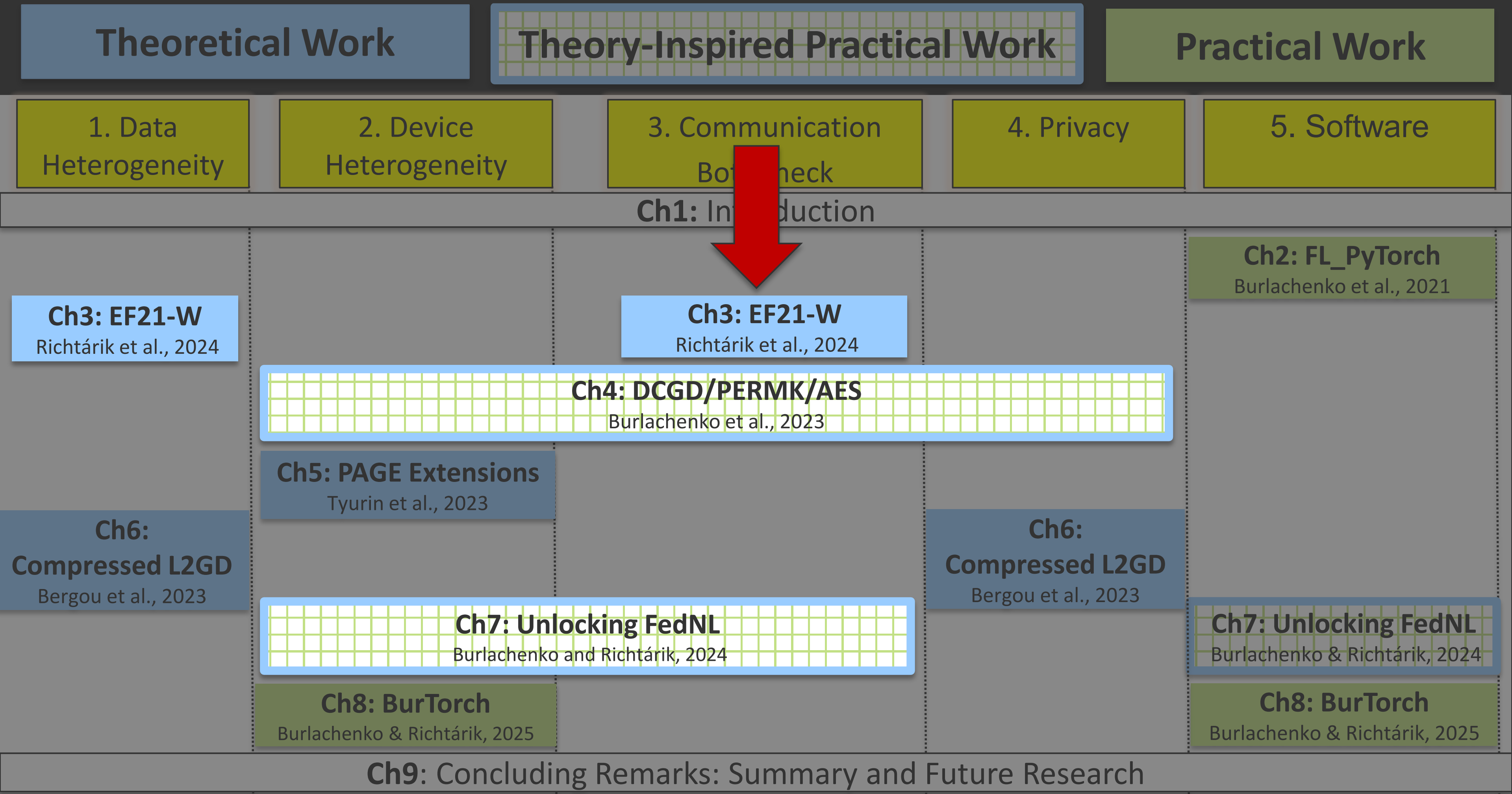
The first publication with “Federated Learning” in its title

While FL mitigates sample size limitations and enables novel decentralized applications, it also brings new challenges

Federated Learning Challenges Addressed in the Thesis #8



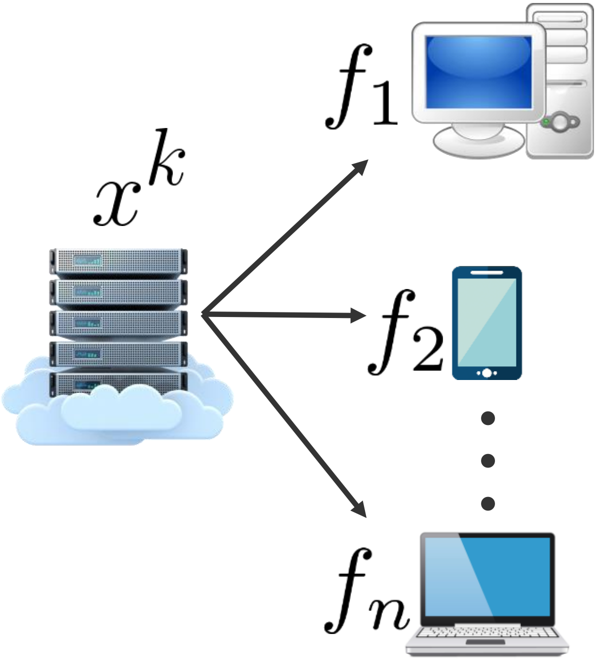
Federated Learning Challenges Addressed in the Thesis



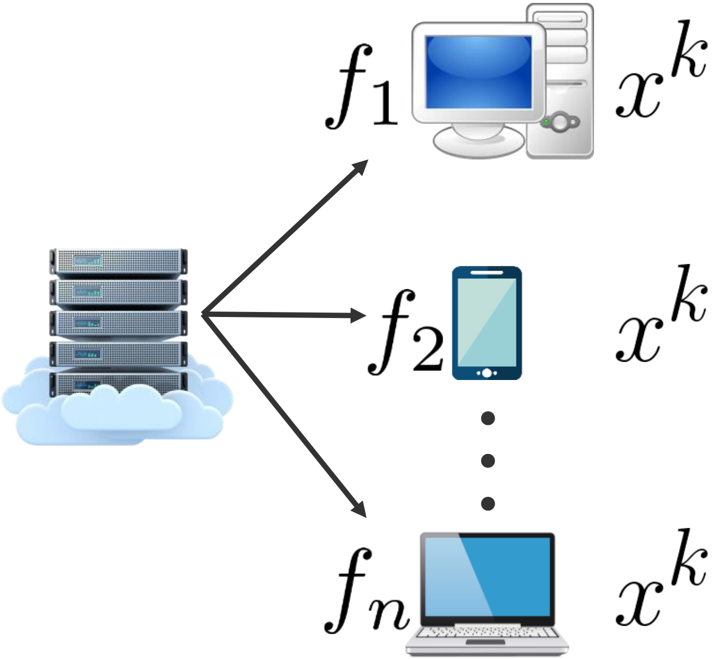
Distributed Gradient Descent

x^k

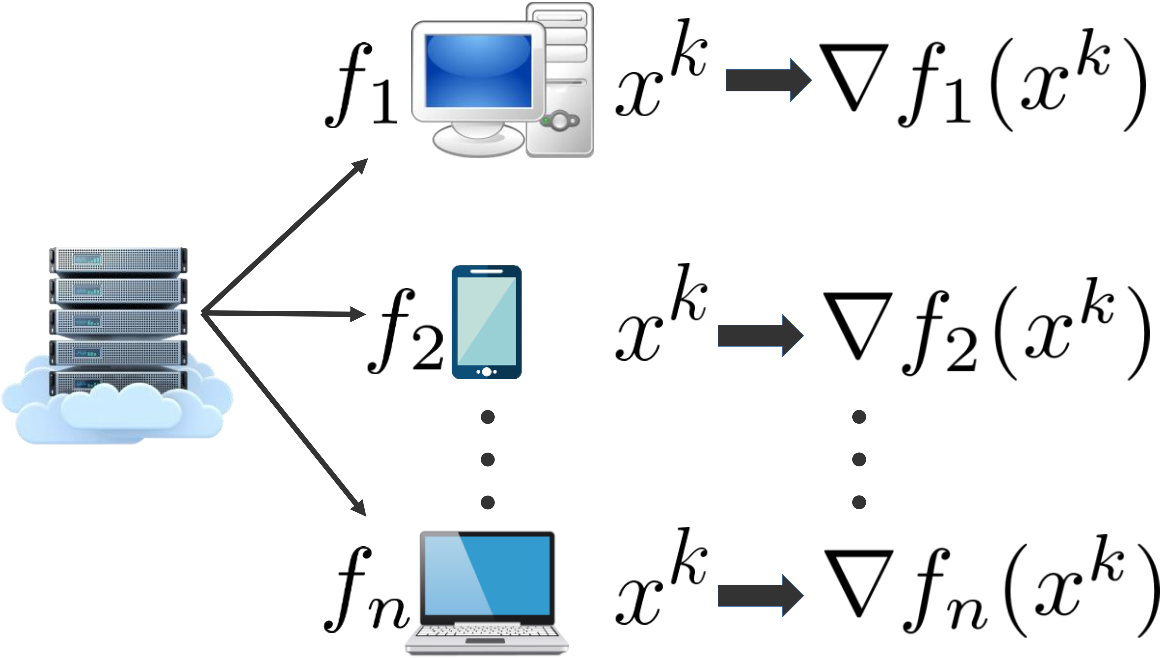
Distributed Gradient Descent



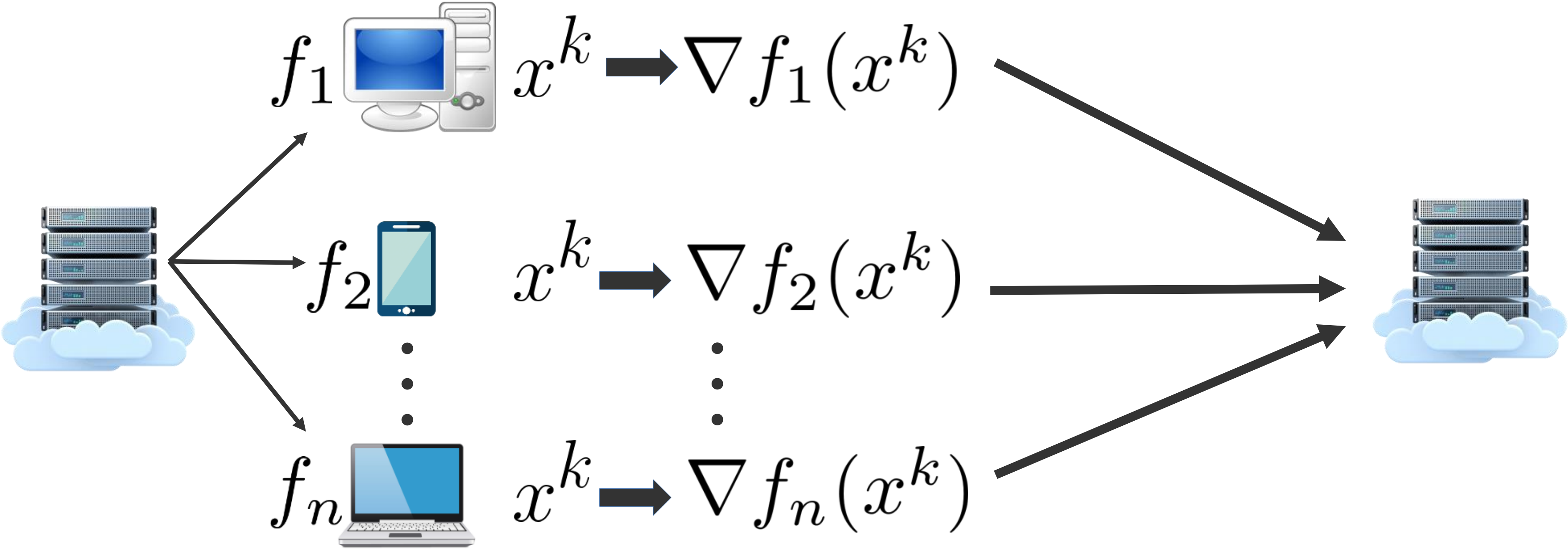
Distributed Gradient Descent



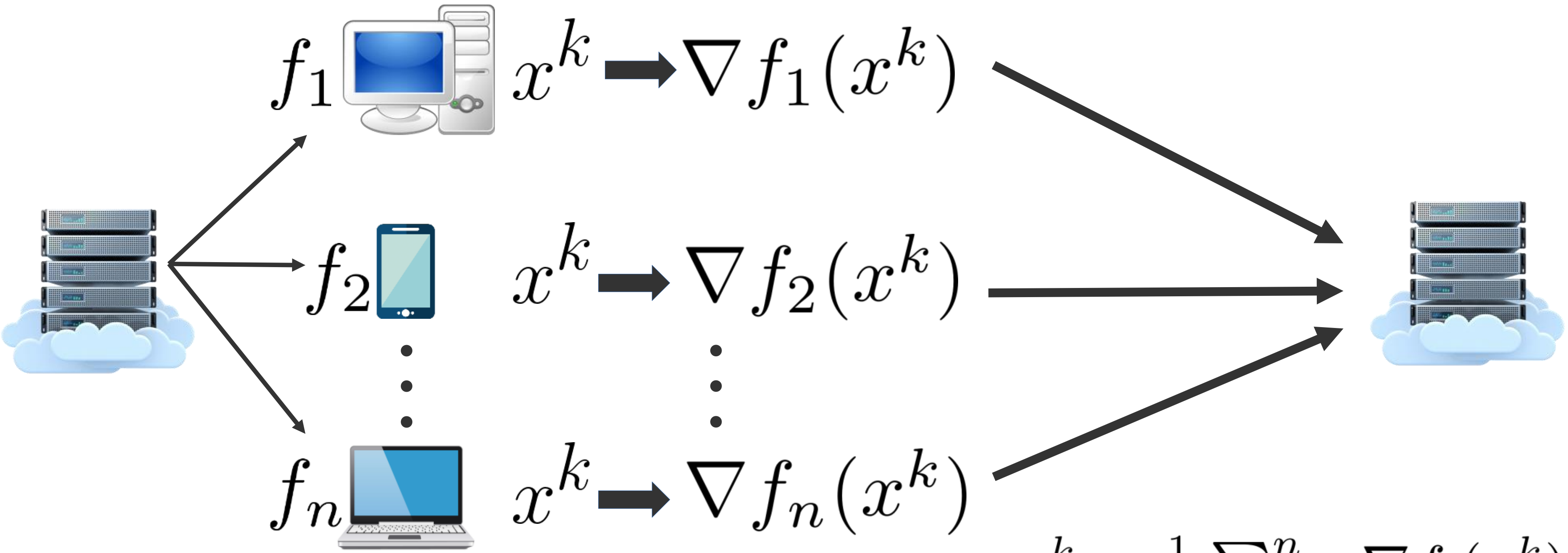
Distributed Gradient Descent



Distributed Gradient Descent



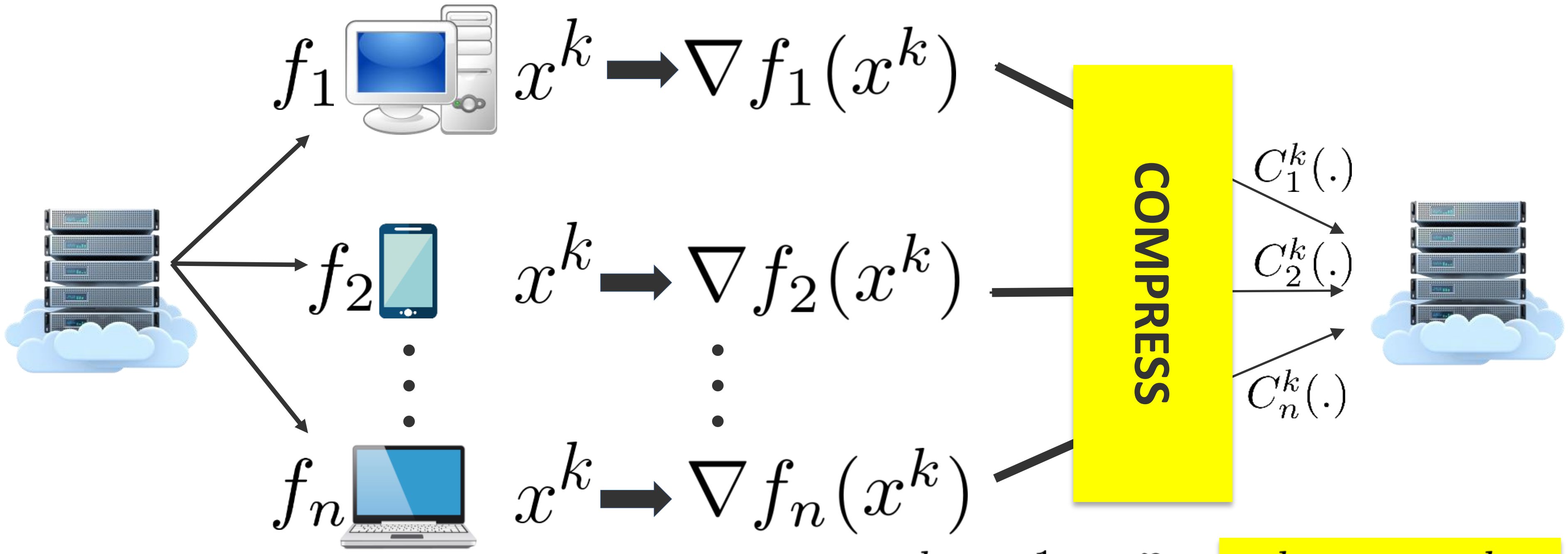
Distributed Gradient Descent



$$g^k = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^k)$$

$$x^{k+1} = x^k - \gamma^k g^k$$

Distributed Compressed Gradient Descent



$$g^k = \frac{1}{n}\sum_{i=1}^{n} C_i^k(\nabla f_i(x^k))$$

$$x^{k+1} = x^k - \gamma^k g^k$$

Compressors

Cost Model

$$\text{Communication Complexity} = (\#\text{Rounds}) \times (\#\text{Bits}/\text{Round})$$

Cost Model

$$\text{Communication Complexity} = (\#\text{Rounds}) \times (\#\text{Bits/Round})$$

Class of Unbiased Compressors

$$\mathcal{B}^d(\omega) = \{B \mid B : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|B(x) - x\|^2\right] \le \omega \|x\|^2, \quad \mathbb{E}\left[B(x)\right] = x\}$$

$\omega \geq 0$

$\forall x \in \mathbb{R}^d$

Compressors

Cost Model

$$\text{Communication Complexity} = (\#\text{Rounds}) \times (\#\text{Bits}/\text{Round})$$

Class of Unbiased Compressors

$$\mathcal{B}^d(\omega) = \{B \mid B : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|B(x) - x\|^2\right] \le \omega \|x\|^2, \quad \mathbb{E}\left[B(x)\right] = x\}$$

Class of Contractive Compressors

$$\mathcal{C}^d(\alpha) = \{C \mid C : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|C(x) - x\|^2\right] \le (1-\alpha) \|x\|^2\}$$

$0 < \alpha \le 1$

$\forall x \in \mathbb{R}^d$

$$B \in \mathcal{B}^d(\omega) \implies C(x) := \frac{1}{\omega + 1} B(x), \quad C(x) \in \mathcal{C}^d\left(\alpha := \frac{1}{\omega+1}\right)$$

Compressors

Cost Model

$$\text{Communication Complexity} = (\#\text{Rounds}) \times (\#\text{Bits/Round})$$

Class of Unbiased Compressors

$$\mathcal{B}^d(\omega) = \{B \mid B : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|B(x) - x\|^2\right] \le \omega \|x\|^2, \quad \mathbb{E}\left[B(x)\right] = x\}$$

Class of Contractive Compressors

$$\mathcal{C}^d(\alpha) = \{C \mid C : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|C(x) - x\|^2\right] \le (1-\alpha)\|x\|^2\}$$

Sparsification Examples $(d = 3, K = 2)$

Input vector: $x = \begin{pmatrix} -9 \\ +7 \\ 3 \end{pmatrix}$

Rand$K$

$S = \{1, 3\}$

$$B(x) := \frac{d}{K} \sum_{i \in S} x_i e_i$$

$$S \sim_{\text{u.a.r.}} \{Q : Q \in 2^{\{1,\dots,d\}} \wedge |Q| = K\}$$

$$\omega = \frac{d}{K} - 1 = \frac{1}{2}$$

Rand$K$: $\begin{pmatrix} -9 \\ 0 \\ 3 \end{pmatrix} \cdot \frac{3}{2}$

Top$K$

$S = \{1, 2\}$

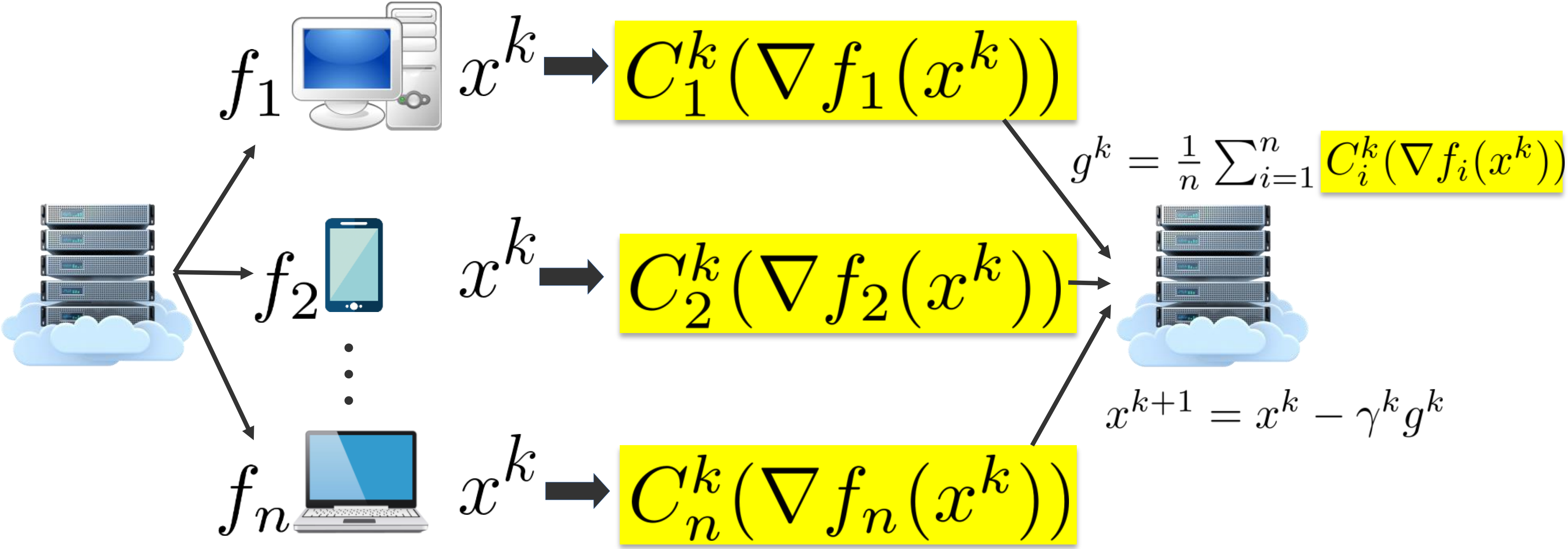
$$C(x) := \sum_{i \in S} x_i e_i$$

$$\alpha = \frac{K}{d} = \frac{2}{3}$$

Top$K$: $\begin{pmatrix} -9 \\ +7 \\ 0 \end{pmatrix}$

Distributed Compressed Gradient Descent With Contractive Compressors

#13



Distributed Compressed Gradient Descent with TopK

leads to exponential divergence even in strongly convex settings ($n = d = 3$)

On Biased Compression for Distributed Learning (2023) Beznosikov et al. (Section 5.2)

EF21

EF21 (Richtárik et al., 2021) is the theoretically fastest method that is provably correct when using contractive compressors

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

Assumptions:

1. $f_i(x)$ are L_i-smooth, but can be non-convex
2. $f(x)$ is L-smooth, but can be non-convex
3. $\exists f^\star > \infty$, such that $f(x) \geq f^\star, \forall x \in \mathbb{R}^d$

Goal:

Find $\widehat{x}$: $\mathbb{E}\left[\|\nabla f(\widehat{x})\|^2\right] \leq \varepsilon^2$

EF21 (Richtárik et al., 2021) is the theoretically fastest method that is provably correct when using contractive compressors

Number of machines

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\| \quad \forall x, y \in \mathbb{R}^d$$

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

Assumptions:

1. $f_i(x)$ are L_i-smooth, but can be non-convex
2. $f(x)$ is L-smooth, but can be non-convex
3. $\exists f^\star > \infty$, such that $f(x) \geq f^\star, \forall x \in \mathbb{R}^d$

Goal:

Find $\widehat{x}$: $\mathbb{E}\left[\|\nabla f(\widehat{x})\|^2\right] \leq \varepsilon^2$

EF21

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n}\sum_{i=1}^n g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

At Client

At Master

EF21

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n} \sum_{i=1}^{n} g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

Total Number of Clients

Iteration

Client

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n} \sum_{i=1}^{n} g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

Communicated from Master to Client

Communicated from Client to Master

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n} \sum_{i=1}^n g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

Reconstructible at the server from
1) received compressed messages
2) previous server states $g_1^{k-1}, \ldots, g_n^{k-1}$

EF21

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n}\sum_{i=1}^n g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

The EF21 analysis allows step size

$$0 < \gamma \leq \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^n L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$

$\approx \frac{1}{\alpha}, \alpha \in (0, 0.5)$

EF21 guarantees

$$\mathbb{E}\left[\|\nabla f(\hat{x}^T)\|^2\right] \leq \frac{2(f(x^0) - f^\star)}{\gamma T} + \frac{G^0}{\theta(\alpha) T}$$

$\theta(\alpha) := 1 - \sqrt{1-\alpha}, \beta(\alpha) := \frac{1-\alpha}{1-\sqrt{1-\alpha}}$

$\alpha = \frac{K}{d}$ for Top$K$ compressor

EF21

EF21: Error Feedback 2021

$$x^{k+1} = x^k - \frac{\gamma}{n}\sum_{i=1}^n g_i^k$$

$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

The EF21 analysis allows step size

$$0 < \gamma \leq \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^n L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}}\right)^{-1}$$

$$f^\star = \inf_{x\in\mathbb{R}^d} f(x)$$

$$G^0 := \sum_{i=1}^n \frac{1}{n}\|g_i^0 - \nabla f_i(x^0)\|^2$$

$$\mathbb{E}\left[\|\nabla f(\hat{x}^T)\|^2\right] \leq \frac{2(f(x^0)-f^\star)}{\gamma T} + \frac{G^0}{\theta(\alpha)T}$$

$$\theta(\alpha) := 1 - \sqrt{1-\alpha}, \beta(\alpha) := \frac{1-\alpha}{1-\sqrt{1-\alpha}}$$

$$\widehat{x}^T \sim_{\text{u.a.r.}} \{x^0, \ldots, x^{T-1}\}$$

Total Iterations

$$\alpha = \frac{K}{d} \text{ for Top}K \text{ compressor}$$

The best step size for EF21

$$\gamma = \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$

The best step size for EF21

$$\gamma = \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$

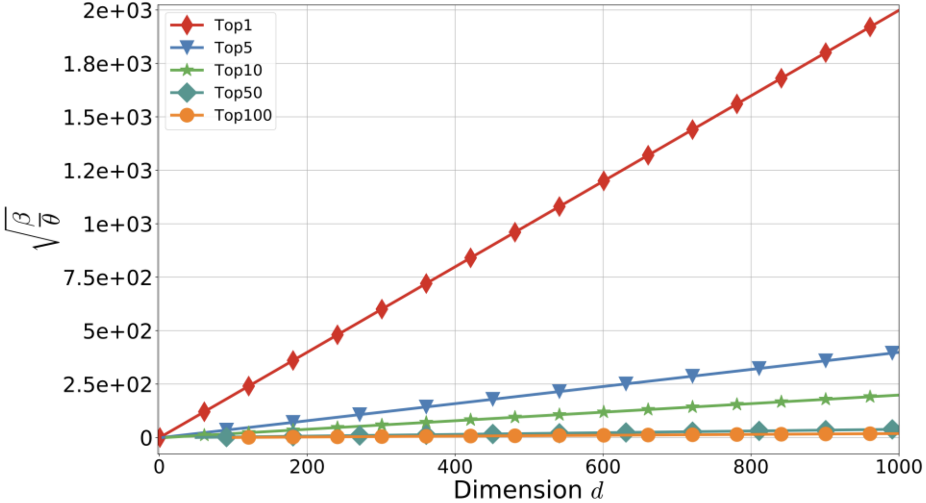
Can we decrease it?

This is already very important

$$L \leq \underbrace{\frac{1}{n}\sum L_i}_{\text{AM}} \leq \underbrace{\sqrt{\frac{1}{n}\sum L_i^2}}_{\text{QM}}$$

The best step size for EF21

$$\gamma = \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$



Can we decrease it?

And it can be arbitrarily big for Top$K$ with $K \ll d$

Ch3: EF21 Reloaded (2024)

The best step size for EF21

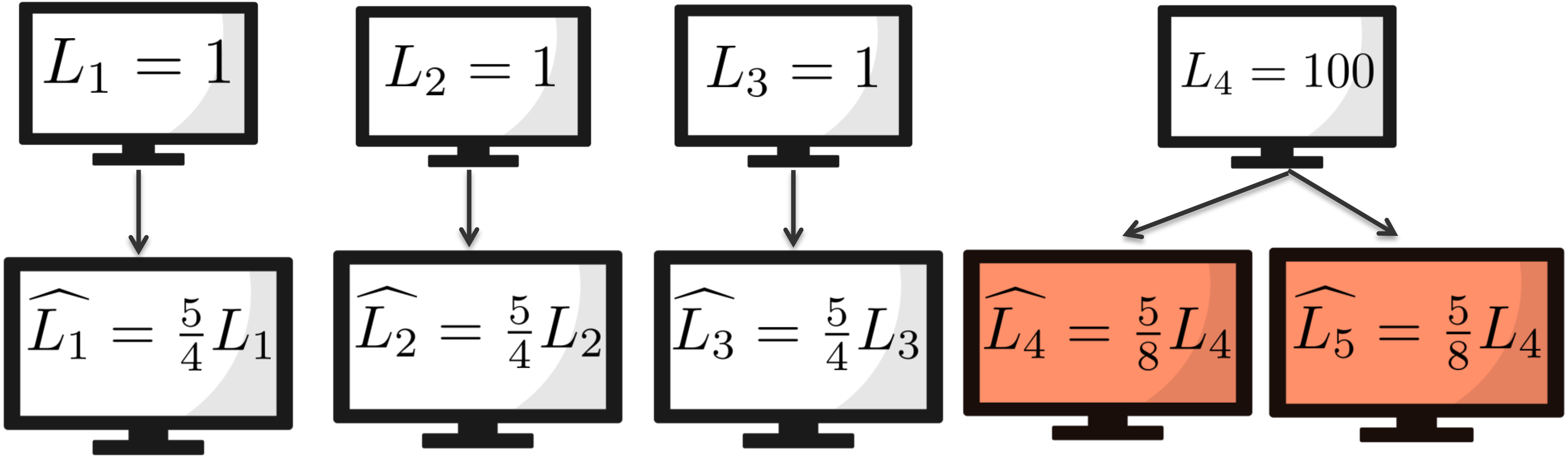
$$\gamma = \left(L + \sqrt{\frac{1}{n}\sum_{i=1}^{n} L_i^2} \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$

We improved the step size in 3 different ways to

$$\gamma = \left(L + \frac{1}{n}\sum_{i=1}^{n} L_i \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1}$$

Example-Driven Reformulation

$$L_{\mathrm{AM}} := \frac{(1+1+1+100)}{4} = 25.75 \qquad L_{QM} = \sqrt{\frac{(1+1+1+100\cdot 100)}{4}} = \sqrt{2500.75}$$



$$\widehat{L}_{\mathrm{AM}} = \frac{3\cdot(5/4)+2\cdot(500/8)}{5} = 25.75 \qquad \widehat{L}_{\mathrm{QM}} = \sqrt{\frac{3\cdot(5/4)^2+2\cdot(500/8)^2}{5}} = \sqrt{1563}$$

QM changed, even AM is the same !

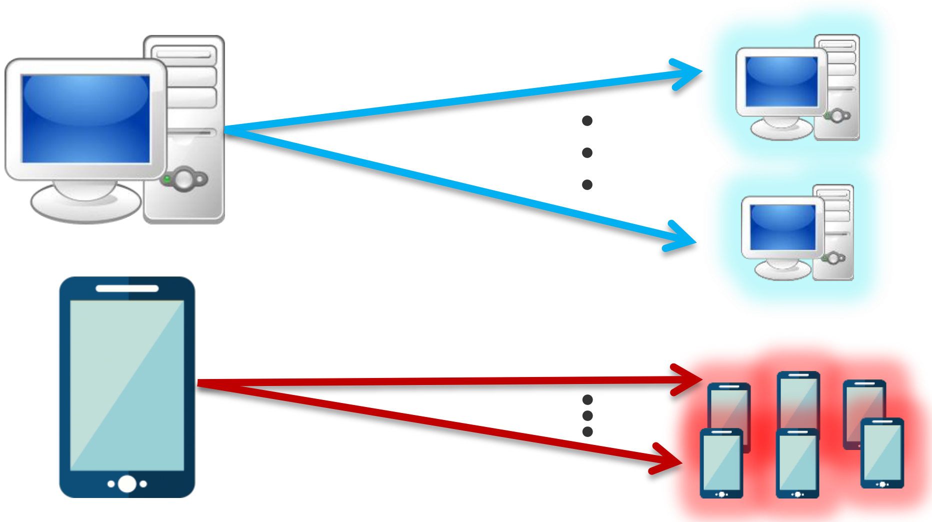
Ch3: EF21 Reloaded (Approach 1)

n clients with $f_i(x)$

$f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$

VIRTUAL CLONING OF EF21 CLIENTS

Ch3: EF21 Reloaded (Approach 1)



n clients with $f_i(x)$

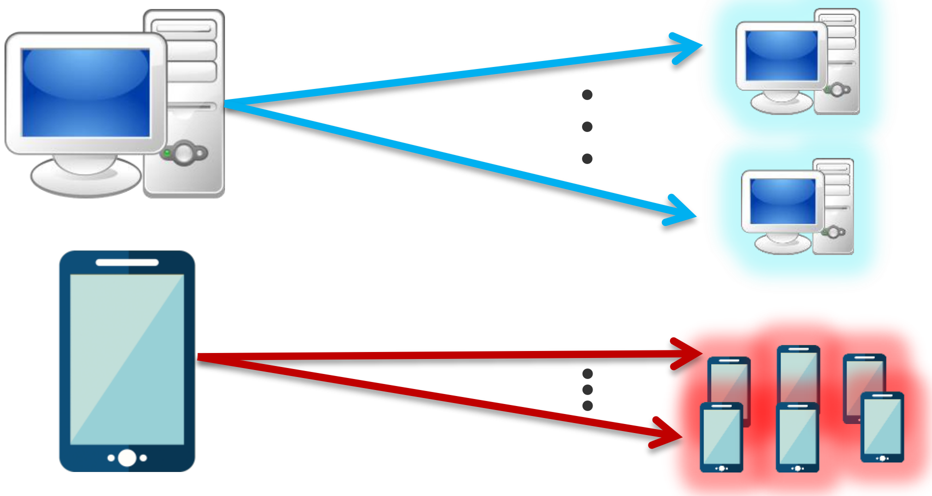
$f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)$

Client i cloned N_i times

$N := \sum_{i=1}^{n} N_i$

VIRTUAL CLONING OF EF21 CLIENTS

Ch3: EF21 Reloaded (Approach 1)



$$\widehat{f_{ij}}(x) = \frac{N}{nN_i} f_i(x)$$

$$\Longrightarrow \quad L_{ij} = \frac{N}{nN_i} L_i$$

n clients with $f_i(x)$

Client i cloned N_i times

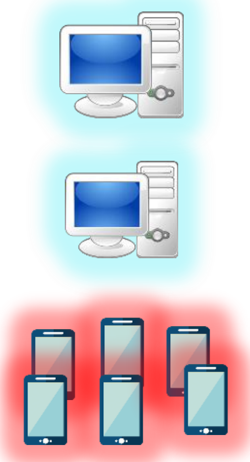
$$f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

$$N := \sum_{i=1}^{n} N_i$$

$$\widehat{f}(x) := \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{N_i} \widehat{f_{ij}}(x)$$

VIRTUAL CLONING OF EF21 CLIENTS

Ch3: EF21 Reloaded (Approach 1)



$$\widehat{f}(x) := \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{N_i}\widehat{f_{ij}}(x)$$

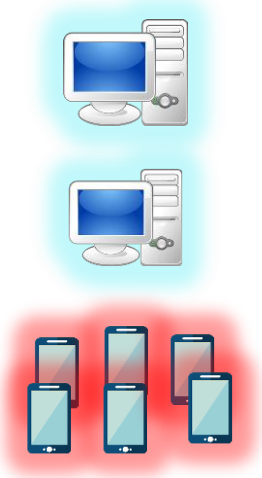
$$\widehat{f_{ij}}(x) = \frac{N}{nN_i} f_i(x),\ L_{ij} = \frac{N}{nN_i} L_i$$

$$N := \sum_{i=1}^{n} N_i$$

Each client i cloned N_i times

VIRTUAL CLONING OF EF21 CLIENTS

Ch3: EF21 Reloaded (Approach 1)



$$\widehat{f}(x) := \frac{1}{N}\sum_{i=1}^{n} \sum_{i=1}^{N_i} \widehat{f_{ij}}(x)$$

$$L_{ij} = \frac{N}{nN_i} L_i$$

$$M(N_1,\dots,N_n) := \sqrt{\frac{1}{N}\sum_{i=1}^{N} \sum_{j=1}^{N_i} L_{ij}^2} = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \frac{L_i^2}{N_i/N}}$$

$$\min_{N_i\in\mathbb{R},\, N_i>0,\, \sum_{i=1}^n N_i/N=1} M(N_1,\dots,N_n) = \frac{\sum_{i=1}^n L_i}{n}$$

$$\frac{\sum_{i=1}^n L_i}{n} \leq M(\lceil L_1/L_{\rm AM}\rceil,\dots,\lceil L_n/L_{\rm AM}\rceil) \leq \left(\frac{1}{n}\sum_{i=1}^n L_i\right)\sqrt{2}$$

$$N_1^\star,\dots,N_n^\star$$

Ch3: EF21 Reloaded (Approach 2)

Good: We reduced QM to AM (up to the factor $\sqrt{2}$)

Bad: We need to increase number of workers $n \to N, n \leq N \leq 2n$

$$\gamma \approx \left(L + \frac{1}{n}\sum_{i=1}^{n} L_i \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1} \qquad N \geq n$$

EF21-W

Good: We reduced QM to AM (up to the factor $\sqrt{2}$)

Bad: We need to increase number of workers $n \to N, n \le N \le 2n$

$$\gamma \approx \left(L + \frac{1}{n}\sum_{i=1}^{n} L_i \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}}\right)^{-1} \qquad N \ge n$$

Assumptions:

B1. Initial shifts for all clones are identical

B2. The compressors are deterministic

⇒ Under these assumptions, the cloning mechanism can be reformulated as a new **EF21-W**

EF21-W

Good: We reduced QM to AM (up to the factor $\sqrt{2}$)

Bad: We need to increase number of workers $n \to N, n \le N \le 2n$

$$\gamma \approx \left(L + \frac{1}{n}\sum_{i=1}^{n} L_i \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1} \qquad N \ge n$$

Assumptions:

B1. Initial shifts for all clones are identical

B2. The compressors are deterministic

⇒ Under these assumptions, the cloning mechanism can be reformulated as a new **EF21-W**

$$x^{k+1} = x^k - \frac{\gamma}{n}\sum_{i=1}^{n} g_i^k$$
$$g_i^{k+1} = g_i^k + C_i^k(\nabla f_i(x^{k+1}) - g_i^k)$$

$$x^{k+1} = x^k - \frac{\gamma}{N}\sum_{i=1}^{n}\sum_{j=1}^{N_i} g_{ij}^k$$
$$g_{ij}^{k+1} = g_{ij}^k + C_i^k(\nabla f_{ij}(x^{k+1}) - g_{ij}^k)$$

$$x^{k+1} = x^k - \gamma \sum_{i=1}^{n} w_i g_i^k$$
$$g_i^{k+1} = g_i^k + C_i^k\left(\frac{1}{n w_i}\nabla f_i(x^{k+1}) - g_i^k\right)$$
$$w_i := \frac{L_i}{\frac{1}{n}\sum_{i=1}^{n} L_i}$$

EF21-W

Good: We reduced QM to AM (up to the factor $\sqrt{2}$)

Bad: We need to increase number of workers $n \to N, n \leq N \leq 2n$

$$\gamma \approx \left(L + \frac{1}{n}\sum_{i=1}^{n} L_i \times \sqrt{\frac{\beta(\alpha)}{\theta(\alpha)}} \right)^{-1} \qquad N \geq n$$

Assumptions:

B1. Initial shifts for all clones are identical

B2. The compressors are deterministic

⇒ Under these assumptions, the cloning mechanism can be reformulated as a new **EF21-W**

$$x^{k+1} = x^k - \gamma \sum_{i=1}^n w_i g_i^k$$
$$g_i^{k+1} = g_i^k + C_i^k\left(\frac{1}{n w_i}\nabla f_i(x^{k+1}) - g_i^k\right)$$

$$w_i := \frac{L_i}{\frac{1}{n}\sum_{i=1}^n L_i}$$

Our analysis reveals that assumptions (B1) and (B2) are not required

EF21-W

The analysis of **EF21-W** reveals that the original EF21 analysis requires modification for the quantity G^t

$$G_i^t := \|g_i^t - \nabla f_i(x^t)\|^2 \quad G^t := \sum_{i=1}^n \frac{1}{n} G_i^t$$

$$G_i^t := \|g_i^t - \frac{\nabla f_i(x^t)}{n w_i}\|^2 \quad G^t := \sum_{i=1}^n w_i G_i^t$$

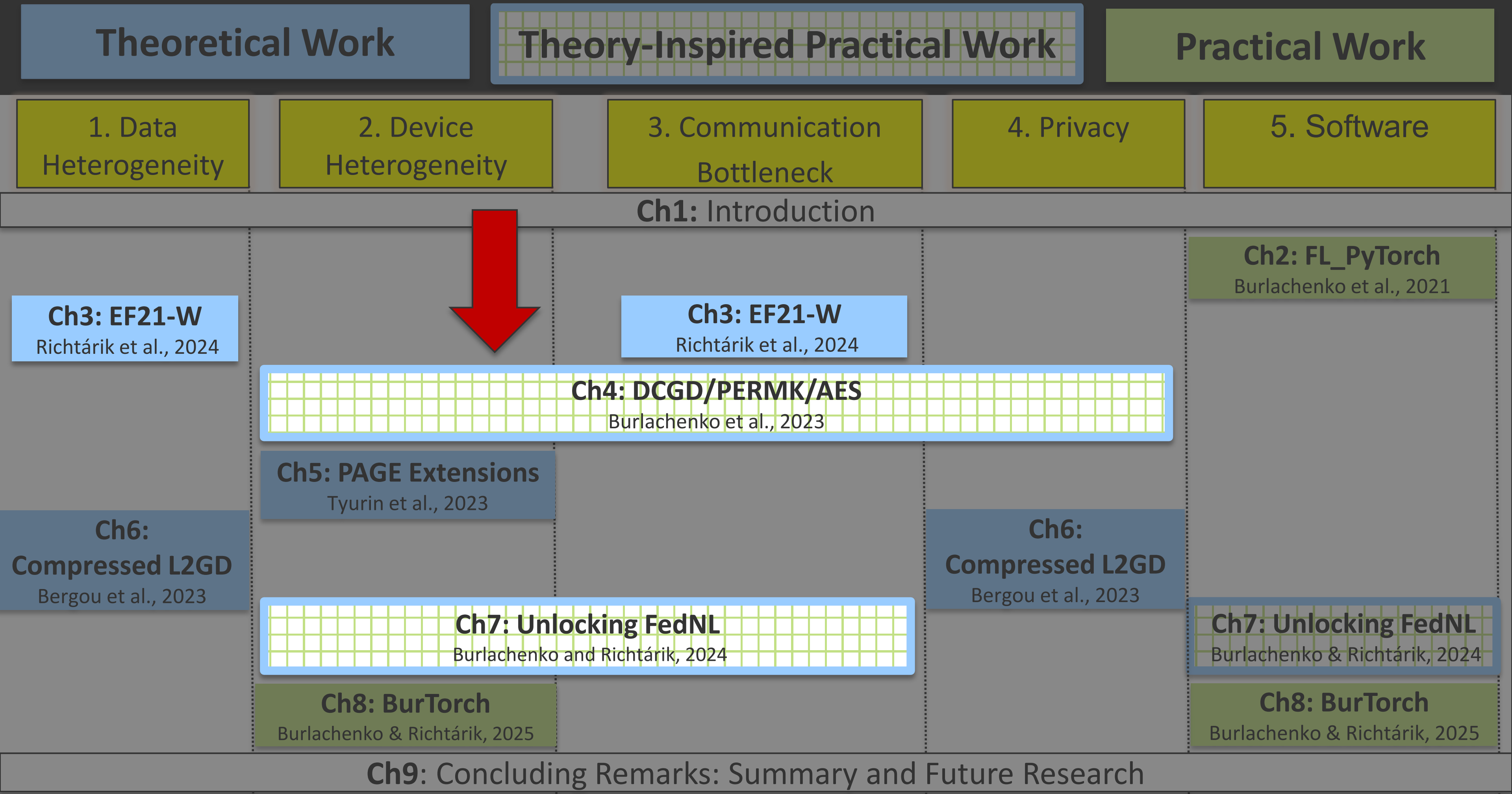
$$w_i := \frac{L_i}{\frac{1}{n}\sum_{i=1}^n L_i}$$

It motivated us to analyze the original EF21 and discover:

Incorporating weights into the original EF21 analysis improves the rate !!

Weights in EF21 Analysis

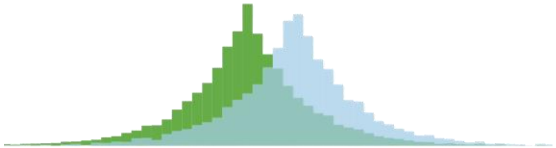
Federated Learning Challenges Addressed in the Thesis



Main Tools for Privacy Guarantees in FL

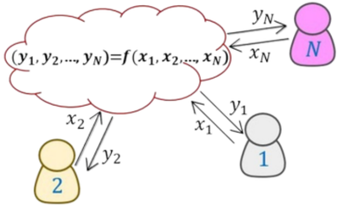
Trusted Execution Environments (TEE)

Protects the execution environment from illegal intervention



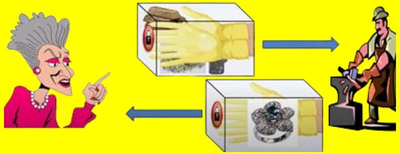
Differential Privacy (DP)

Protects output of algorithm so that users' data are not leaking after execution



Secure Multi-Party Computation (MPC)

Protects inputs of algorithm at the cost of communication



Homomorphic Encryption (HE)

Computation on encrypted data without revealing inputs or outputs

Homomorphic Encryption (HE)

Homomorphism of two groups G_1 and G_2 is a mapping $f : G_1 \to G_2$

$$f(x * y) = f(x) * f(y), \quad \forall x, y \in G_1$$

Homomorphic Encryption:

Computation on encrypted data without revealing inputs or outputs

Homomorphic Encryption In Action:

1. Any device with the **public key** can perform computations on encrypted data
2. Only the holder of the **private key** can decrypt the result

Cheon-Kim-Kim-Song (CKKS, 2017):

a. The **CKKS** scheme supports approximate arithmetics on real and complex dense vectors and is considered as SOTA in this class

b. **CKKS (and HE in general)** is more complex primitive than classical block ciphers (e.g. AES-based), relying on entirely different mathematical foundations

Distributed Compressed Gradient Descent with Homomorphic Encryption (HE)

$$x^k \rightarrow Enc\left(\underbrace{\mathbf{C}_1\left(\nabla f_1(x^k)\right)}_{\text{message}}\right)$$

Average of Encrypted Messages = Encryption of Average of Plain Messages

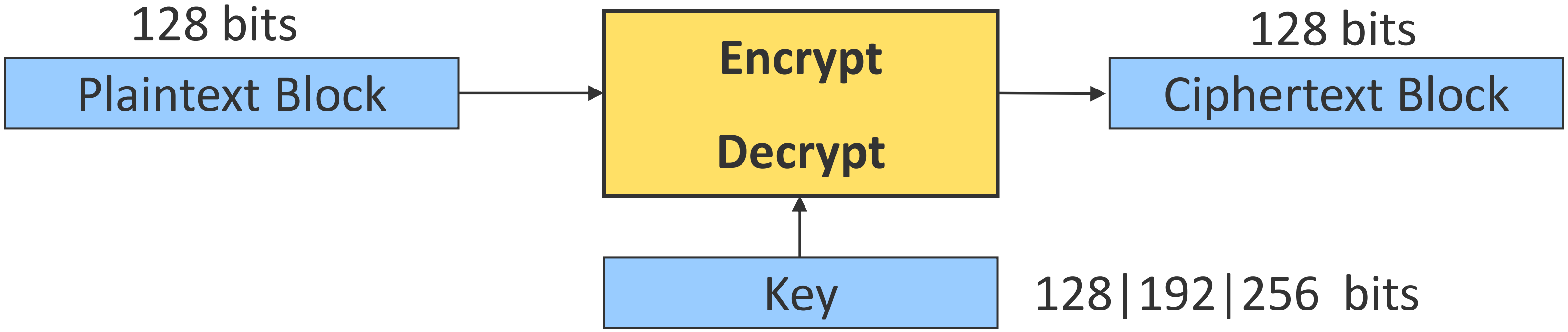
$$x^k \rightarrow Enc\left(\left(C_2(\nabla f_2(x^k)\right)\right)$$

$$x^k \rightarrow Enc(C_n(\nabla f_n(x^k)))$$

$$g^k = \frac{1}{n}\sum_{i=1}^n Enc(C_i(\nabla f_i(x^k))) = Enc\left(\frac{1}{n}\sum_{i=1}^n C_i(\nabla f_i(x^k))\right)$$

$$x^{k+1} = x^k - \gamma^k Dec(g^k)$$

Classical Cryptography: AES Block Cipher



AES (2001) Block Cipher

- Maps deterministically and with reversible operations input (128 bits) into output (128 bits)
- Has hardware support (Intel Westmere, AMD Bulldozer, ARM Cortex-A53)
- AES is a strong cryptographic primitive, widely trusted as a secure **PRP**

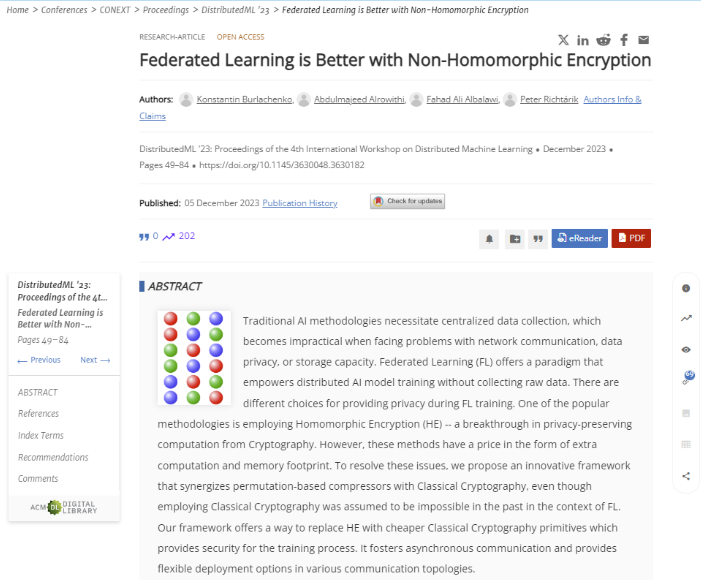
A secure pseudorandom permutation (PRP)
produces permutations that are computationally indistinguishable from uniformly random permutations by any *known* polynomial-time algorithm

DP, HE, MPC, TEE…
But where is Classical Cryptography?

Researchers from 2020 – 2024 consistently argue that applying symmetric-key encryption like AES or DES in FL is unsuitable, challenging, not feasible

Secure, Privacy-Preserving, and Federated Machine Learning in Medical Imaging,
G. Kaissis et al. (**2020**) Nature Machine Intelligence
Private Artificial Intelligence: Machine Learning on Encrypted Data,
Kristin E. Lauter (**2022**) SIAM
Cybersecurity English Accelerated Encrypted Execution of General Purpose Applications,
V. Joseph et al (**2023**) NVIDIA Blog
FedSHE: Privacy-Preserving and Efficient Federated Learning with Adaptive Segmented CKKS Homomorphic Encryption,
Pan Y. et al. (**2024**) Cybersecurity
Revisiting Fully Homomorphic Encryption Schemes for Privacy-Preserving Computing,
N. Jain et al. (**2024**) Emerging Technologies and Security in Cloud Computing

Ch4: DCGD/PermK/AES (2023)


Home > Conferences > CONEXT > Proceedings > DistributedML '23 > Federated Learning is Better with Non-Homomorphic Encryption

RESEARCH-ARTICLE OPEN ACCESS

Federated Learning is Better with Non-Homomorphic Encryption

Authors: Konstantin Burlachenko, Abdulmajeed Alrowithi, Fahad Ali Albalawi, Peter Richtárik Authors Info & Claims

DistributedML '23: Proceedings of the 4th International Workshop on Distributed Machine Learning • December 2023 • Pages 49–84 • https://doi.org/10.1145/3630048.3630182

Published: 05 December 2023 Publication History

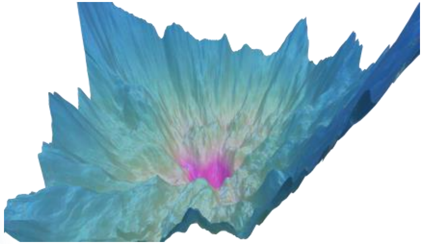
DistributedML '23: Proceedings of the 4t...
Federated Learning is Better with Non-...
Pages 49–84
← Previous Next →

ABSTRACT
References
Index Terms
Recommendations
Comments

ACM DL DIGITAL LIBRARY

ABSTRACT

Traditional AI methodologies necessitate centralized data collection, which becomes impractical when facing problems with network communication, data privacy, or storage capacity. Federated Learning (FL) offers a paradigm that empowers distributed AI model training without collecting raw data. There are different choices for providing privacy during FL training. One of the popular methodologies is employing Homomorphic Encryption (HE) -- a breakthrough in privacy-preserving computation from Cryptography. However, these methods have a price in the form of extra computation and memory footprint. To resolve these issues, we propose an innovative framework that synergizes permutation-based compressors with Classical Cryptography, even though employing Classical Cryptography was assumed to be impossible in the past in the context of FL. Our framework offers a way to replace HE with cheaper Classical Cryptography primitives which provides security for the training process. It fosters asynchronous communication and provides flexible deployment options in various communication topologies.



Distributed **C**ompressed **G**radient **D**escent

Permutated Correlated Compressors

Advanced **E**ncryption **S**tandard

DCGD/PermK

x^k

x^k

x^k

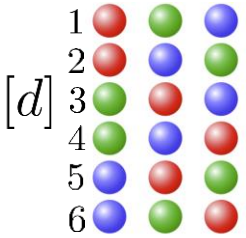
DCGD/PermK

PermK Compressors (Rafał Szlendak, et al. 2021)

$x^k \rightarrow$ $g_i^k = B_i(\nabla f_i(x^k))$

$x^k \rightarrow$

$x^k \rightarrow$



$k = 1, 2, 3, \ldots$

DCGD/PermK

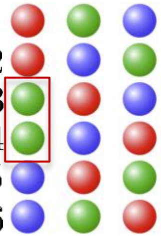
Green user uses coordinates $3, 4$ from $[d] = \{1, \ldots, 6\}$

Example

$x^k \rightarrow g_i^k = B_i(\nabla f_i(x^k))$

$x^k \rightarrow$

$x^k \rightarrow$



$k = 1, 2, 3, \ldots$

$\nabla f_{1,2,3}(x^k)$

0.1	1.1	2.1
0.2	1.2	2.2
0.3	1.3	2.3
0.4	1.4	2.4
-0.5	1.5	2.5
0.6	1.6	2.6

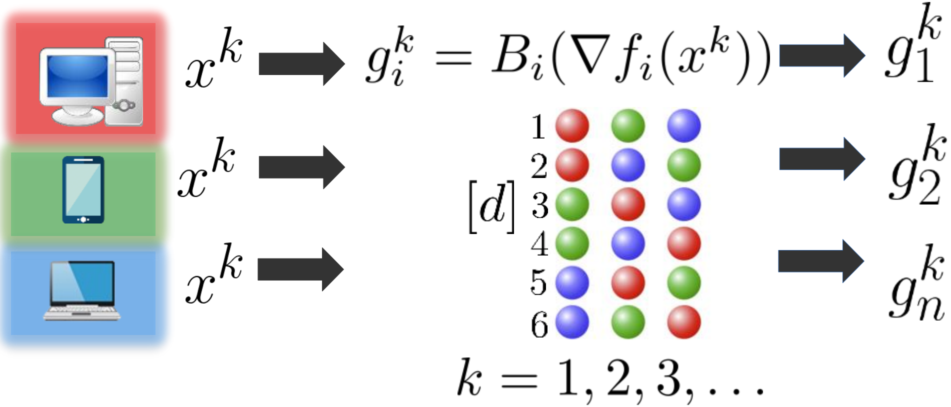
⇨

g_1^k	g_2^k	g_3^k
0.1	0	0
0.2	0	0
$3\times$ 0	$3\times$ 1.3	$3\times$ 0
0	1.4	0
0	0	2.5
0	0	2.6

Training Iteration

Scaling is needed to preserve unbiasedness

DCGD/PermK

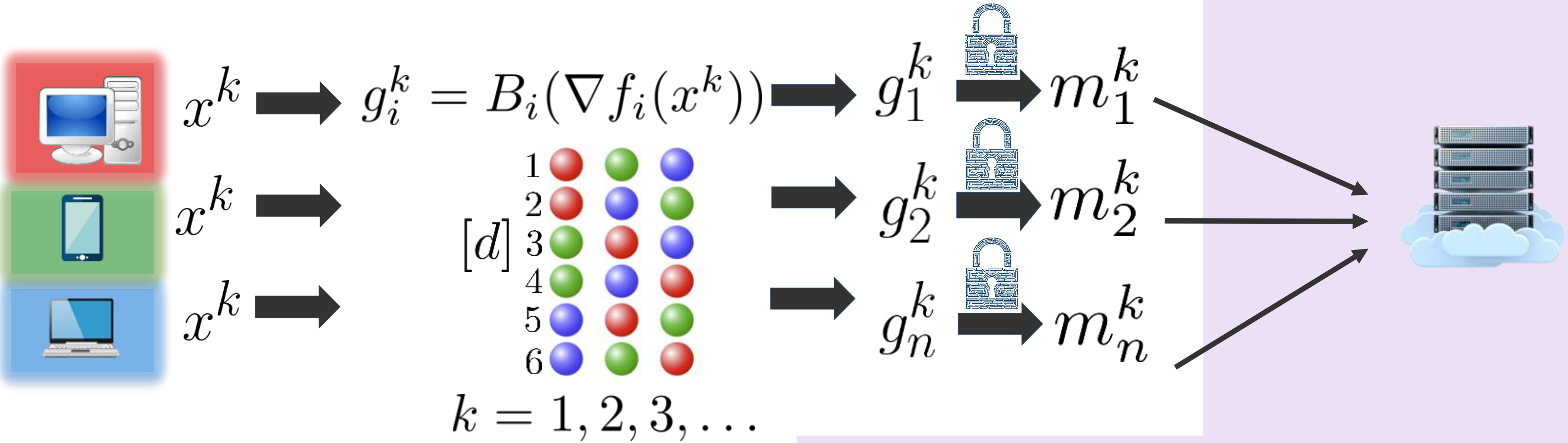


Algebraic Properties of PermK

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n B_i(v_i)\right] = \frac{1}{n}\sum_{i=1}^n v_i \qquad \forall v_i \in \mathbb{R}^d$$

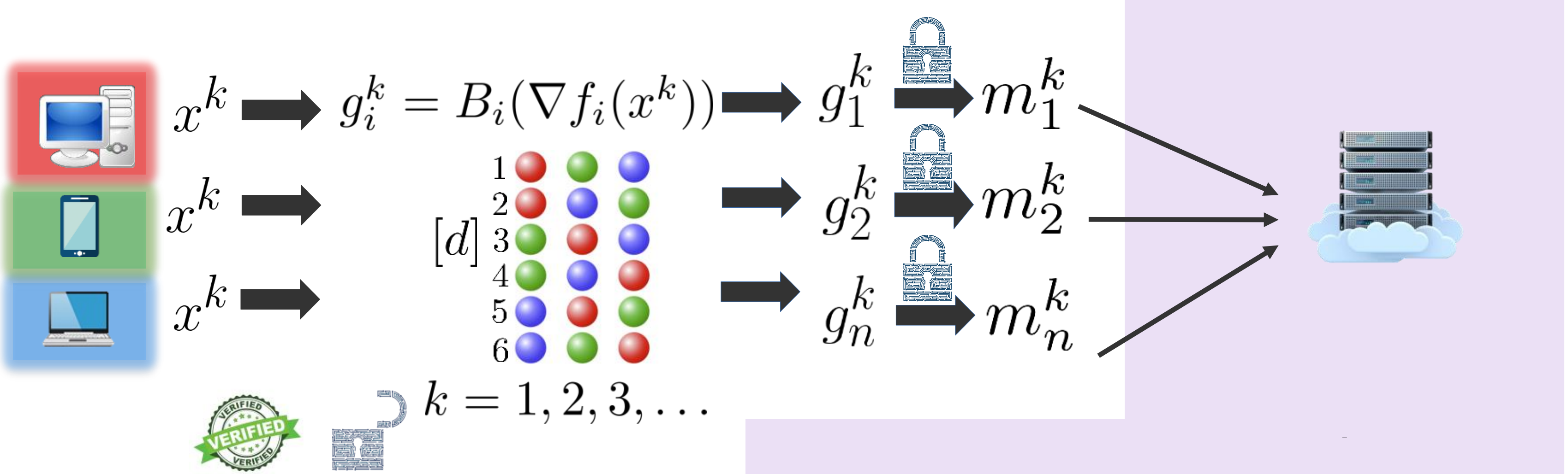
$$\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i=1}^n B_i(v_i) - \frac{1}{n}\sum_{i=1}^n v_i\right\|^2\right] \leq \frac{1}{n}\sum_{i=1}^n \|v_i\|^2 - \left\|\frac{1}{n}\sum_{i=1}^n v_i\right\|^2$$

DCGD/PermK/AES



$$g^k = \sum_{i=1}^n m_i^k = \mathrm{concat}(\mathrm{m}_1^k, \ldots, \mathrm{m}_n^k)$$

DCGD/PermK/AES

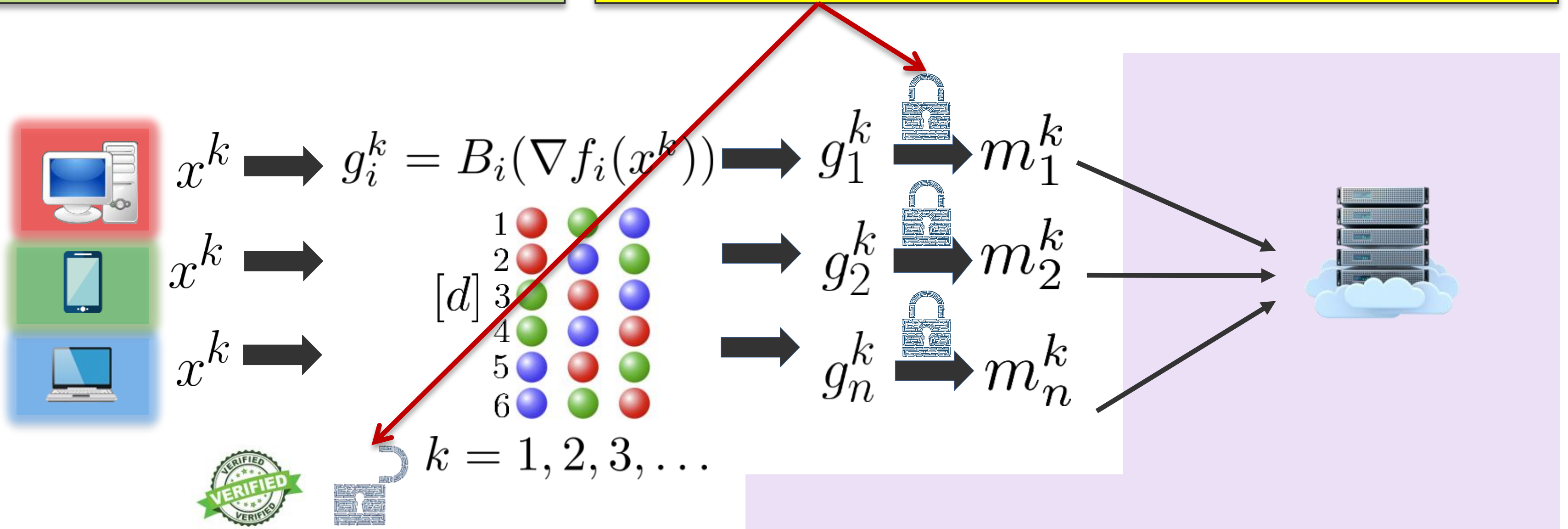


$$x^{k+1} = x^k - \gamma^k \cdot \frac{1}{n}[Dec(m_1^k), \ldots, Dec(m_n^k)]^\top \leftarrow g^k = \sum_{i=1}^{n} m_i^k = \mathrm{concat}(\mathrm{m}_1^k, \ldots, \mathrm{m}_n^k)$$

DCGD/PermK/AES

HE/CKKS for AES-128 security level requires key(s) with sizes 420 000 bytes

For AES-128 the key size is 128 bits (16 bytes)



$$x^{k+1} = x^k - \gamma^k \cdot \frac{1}{n}[Dec(m_1^k), \ldots, Dec(m_n^k)]^\top \leftarrow g^k = \sum_{i=1}^n m_i^k = \mathrm{concat}(\mathrm{m}_1^k, \ldots, \mathrm{m}_\mathrm{n}^k)$$

DCGD/PermK/AES vs HE

$$\text{For } g_i^k = B_i(\nabla f_i(x^k))$$

$$\mathrm{Enc}\left(\frac{1}{n}\sum_{i=1}^{n} g_i^k\right) = \frac{1}{n}\mathrm{Concat}(\mathrm{Enc}(g_1^k), \mathrm{Enc}(g_2^k), \ldots, \mathrm{Enc}(g_n^k))$$

NEW

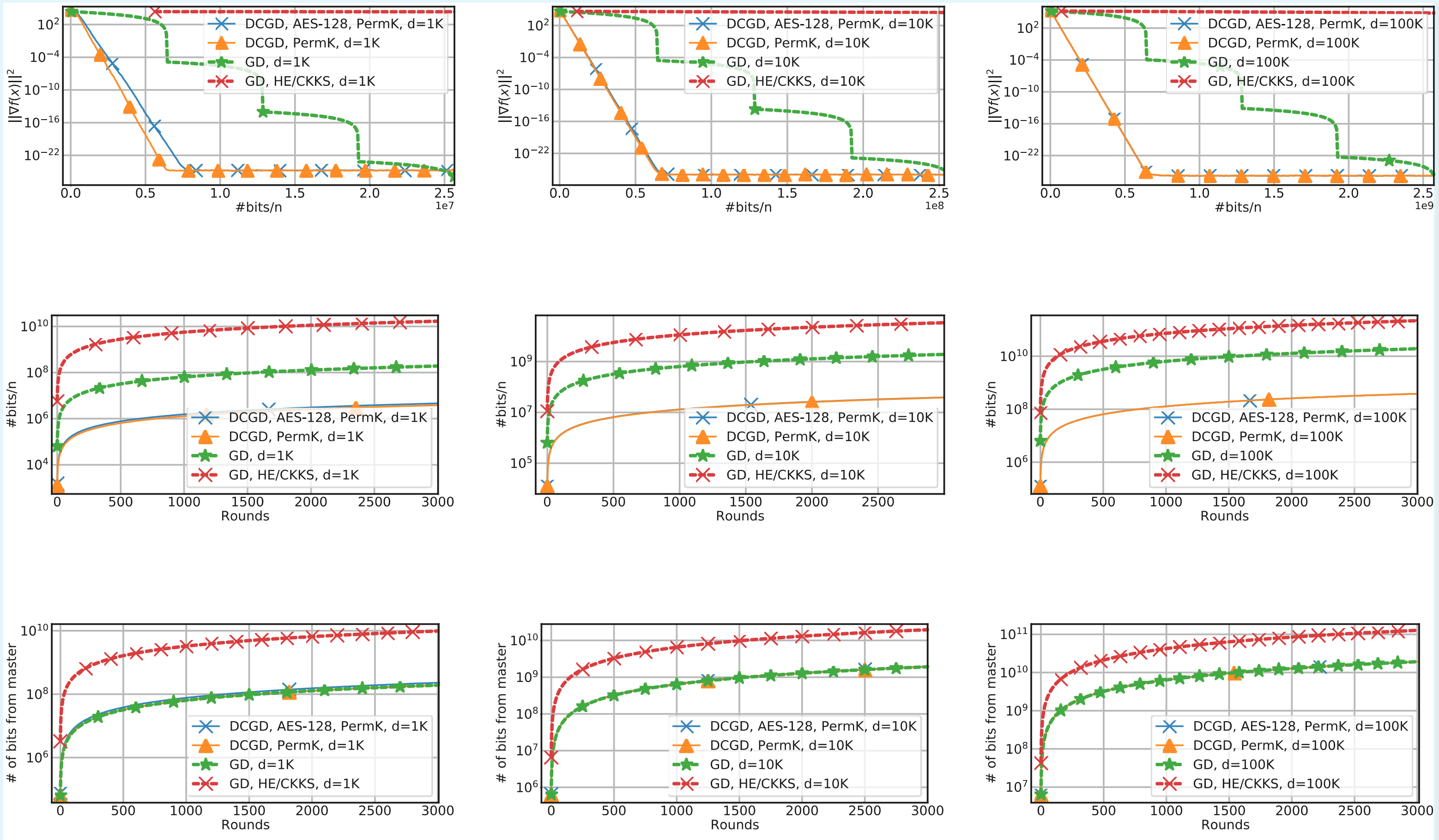
- – Only compatible with specific
- \+ Does not introduce numerical errors
- \+ Low memory overhead from AES

$$\forall g_i^k \in \mathbb{R}^d$$

$$\mathrm{Enc}\left(\frac{1}{n}\sum_{i=1}^{n} g_i^k\right) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Enc}(g_i^k)$$

CLASSICAL HE

- \+ Works with arbitrarily
- – Introduces numerical errors
- – High memory overhead



(a) $d = 1\,000$ (b) $d = 10\,000$ (c) $d = 100\,000$

Linear regression

Interpolation regime

$n = 50,\ n_i = 12$

Compute FP64

Tuned $\gamma = 0.007$ for DCGD

Theoretical γ for GD

Semi-Asynchronous Behavior for DCGD/PermK

$$\min_{x\in\mathbb{R}^d}\left\{f(x) := \frac{1}{4}\sum_{i=1}^{4} f_i(x)\right\}$$

Gradient Descent

$g^k = \sum_{i=1}^4 \nabla f_i(x)$

$x^{k+1} = x^k - \gamma \cdot \frac{1}{n} \cdot g^k$

Requires synchronization among clients

DCGD/PermK/AES

$g^k = [p_1, p_2, p_3, p_4]^\top$

$x^{k+1}_{\text{parts:2,3,4}} = x^k_{\text{parts:2,3,4}} - \gamma \cdot \frac{1}{n} \cdot [g^k]_{\text{parts:2,3,4}}$

Start forward pass for next iteration with partial x^{k+1}

Wait for straggler (client number #1)

Federated Learning Challenges Addressed in the Thesis

Theoretical Work | **Theory-Inspired Practical Work** | **Practical Work**

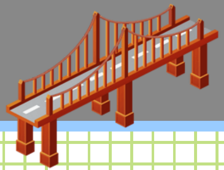
1. Data Heterogeneity
2. Device Heterogeneity
3. Communication Bottleneck
4. Privacy
5. Software

Ch1: Introduction

Ch2: FL_PyTorch
Burlachenko et al., 2021

The $f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ holds a surprising property

Ch3: EF21-W
Richtárik et al., 2024



Ch3: EF21-W
Richtárik et al., 2024

Ch4: DCGD/PERMK/AES
Burlachenko et al., 2023

Ch5: PAGE Extensions
Tyurin et al., 2023

Ch6: Compressed L2GD
Bergou et al., 2023

Ch6: Compressed L2GD
Bergou et al., 2023

Ch7: Unlocking FedNL
Burlachenko and Richtárik, 2024

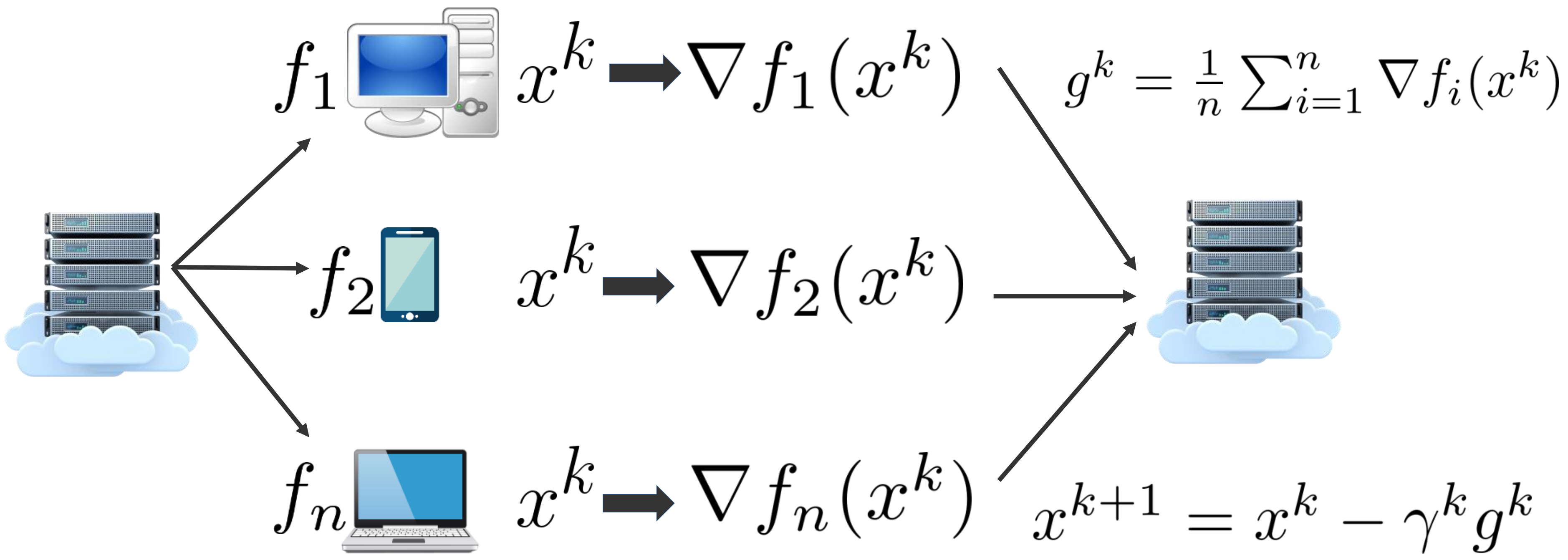
Ch7: Unlocking FedNL
Burlachenko & Richtárik, 2024

Ch8: BurTorch
Burlachenko & Richtárik, 2025

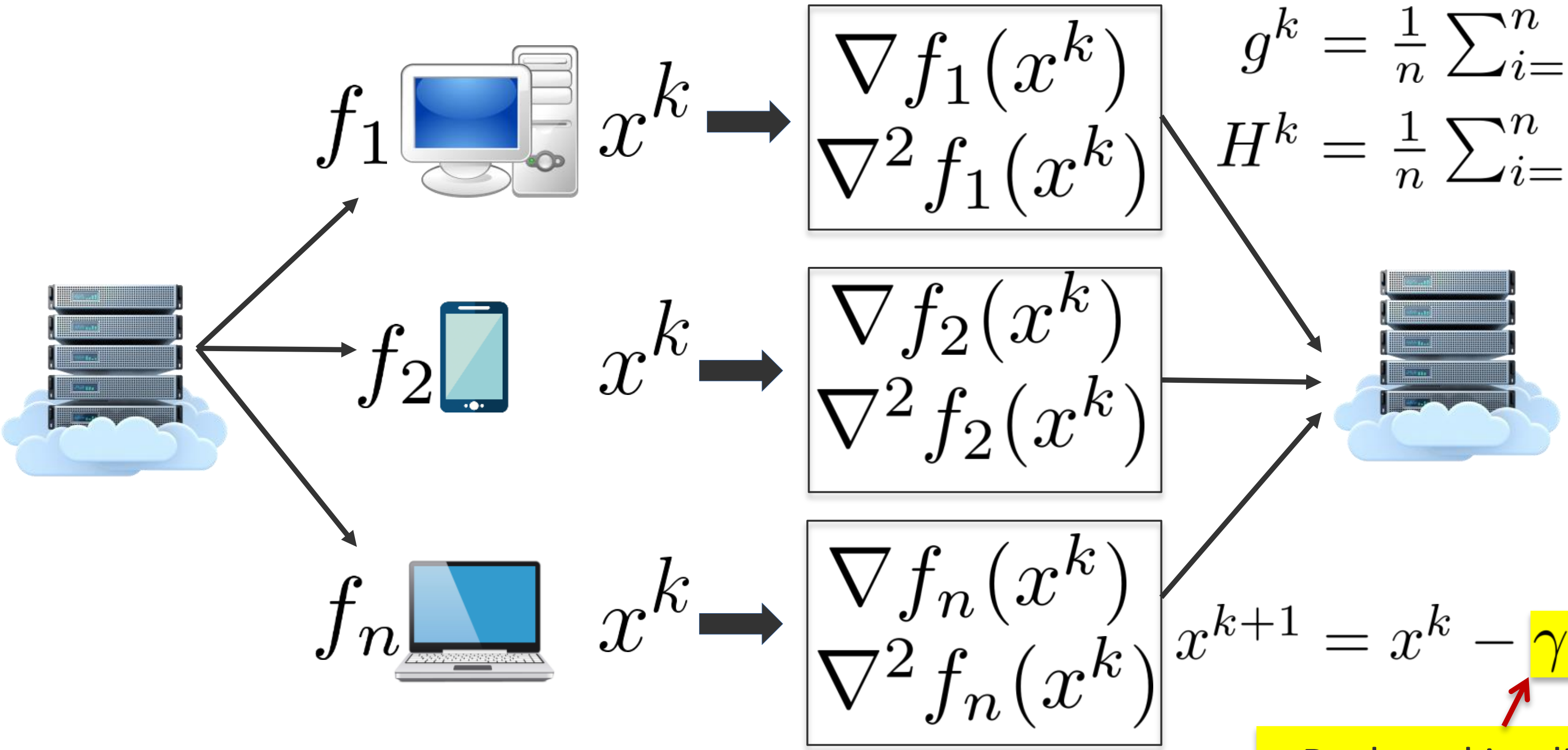
Ch8: BurTorch
Burlachenko & Richtárik, 2025

Ch9: Concluding Remarks: Summary and Future Research

Distributed Gradient Descent => Distributed Newton



Distributed Gradient Descent => Distributed Newton



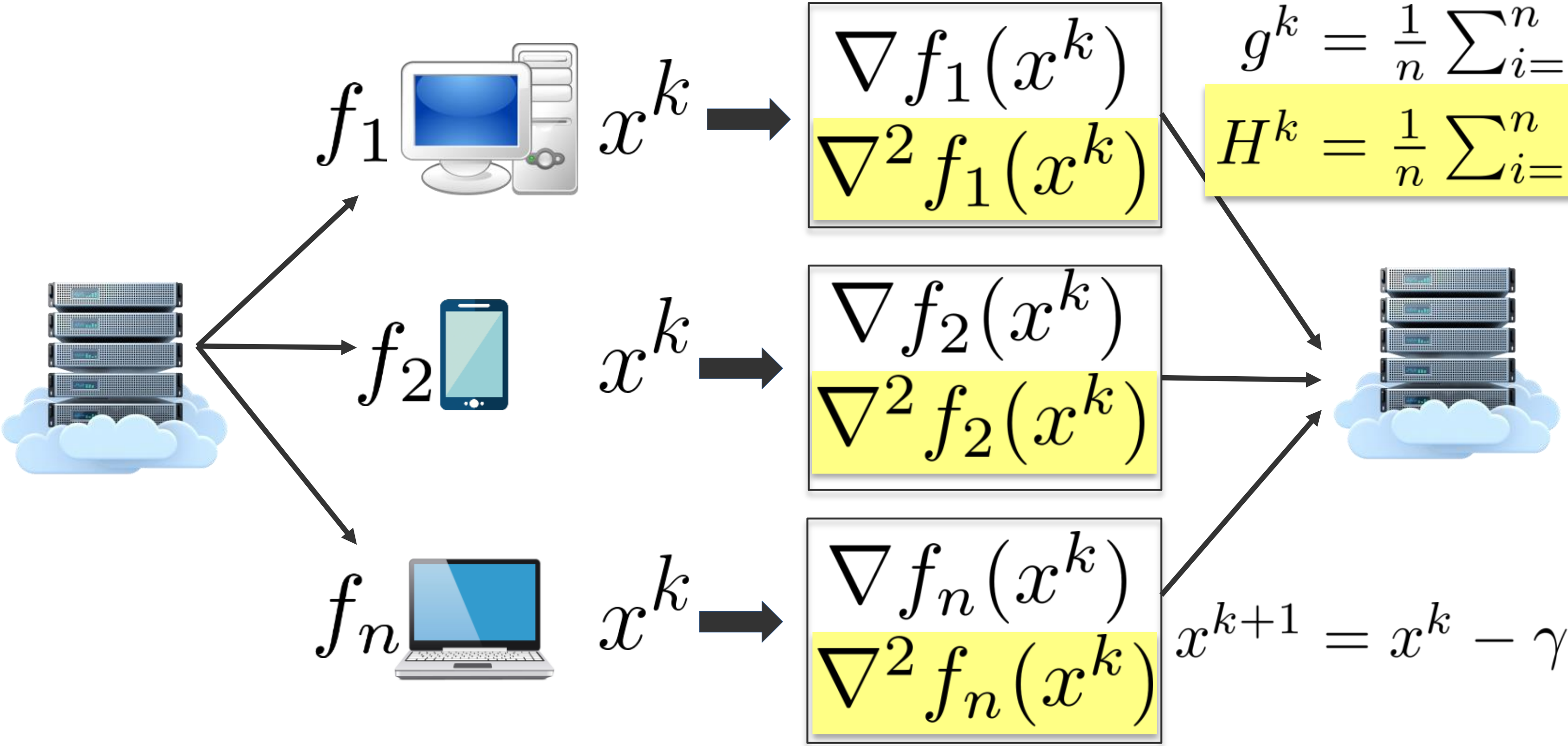
$$g^k = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^k)$$

$$H^k = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(x^k)$$

$$x^{k+1} = x^k - \gamma (H^k)^{-1} g^k$$

Backtracking line search
[BV, 2004]

Distributed Gradient Descent => Distributed Newton



$$g^k = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^k)$$

$$H^k = \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(x^k)$$

$$x^{k+1} = x^k - \gamma (H^k)^{-1} g^k$$

Bad: The memory requirement for forming and storing second-order information

Distributed Gradient Descent => Distributed Newton

$$\mu \cdot I \preceq \nabla^2 f(x) \preceq L \cdot I$$

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq L_* \|x - y\|_2$$

$$\|\nabla f(x^k)\| \leq \eta \implies \gamma = 1, \quad \frac{L_*}{2\mu^2}\|\nabla f(x^{k+1})\|_2 \leq \left(\frac{L_*}{2\mu^2}\|\nabla f(x^k)\|_2\right)^2$$

[BV, 2004]

$$x^k \rightarrow \begin{matrix}\nabla f_n(x^k) \\ \nabla^2 f_n(x^k)\end{matrix}$$

$$x^{k+1} = x^k - \gamma (H^k)^{-1} g^k$$

Good: $\text{Error}^{k+1} \leq \text{Const} \cdot \left(\text{Error}^k\right)^2$. In practice, number of iterations is ≈ 6.

Federated Newton Learn (2022) (Existing)

Problem

$$x^\star = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left(f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right)$$

FedNL: Federated Newton Learn (M. Safaryan et al., 2022)

$$x^{k+1} = x^k - \left[\tfrac{1}{n}\sum_{i=1}^{n} H_i^k + l^k I\right]^{-1} \tfrac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^k)$$

$$H_i^{k+1} = H_i^k + C_i^t(\nabla^2 f_i(x^{k+1}) - H_i^k)$$

EF21 Mechanism

FedNL Technicality

Generalizing Biased and Unbiased Compressors to Symmetric Matrices

Assumptions for FedNL Family

$f(x)$ is μ strongly convex and $f_i(x)$ has Lipschitz continuous Hessian

Federated Newton Learn (2022) (Existing)

Problem

$$x^\star = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left(f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right)$$

FedNL: Federated Newton Learn (M. Safaryan et al., 2022)

$$x^{k+1} = x^k - \left[\tfrac{1}{n}\sum_{i=1}^n H_i^k + l^k I\right]^{-1} \tfrac{1}{n}\sum_{i=1}^n \nabla f_i(x^k)$$

$$H_i^{k+1} = H_i^k + C_i^t(\nabla^2 f_i(x^{k+1}) - H_i^k)$$

FedNL Local Superlinear Convergence Guarantees

D depend on Smoothness of Hessian

$C \in \{2, 8\}$

For $\|C(M) - M\|_F^2 \leq (1-\alpha)\|M\|_F$ the $A = \frac{\alpha}{4}$

$$\|x^0 - x^\star\| \leq \frac{\mu}{\sqrt{2D}} \qquad H^k \leq \frac{\mu^2}{4C}$$

$$H^k := \frac{1}{n}\sum_{i=1}^n \|H_i^k - \nabla^2 f_i(x^\star)\|_F^2$$

$\|\nabla^2 f(x) - \nabla^2 f(y)\|_F \leq L_F \|x - y\|_2$

$$\Phi^k := H^k + 6BL_F^2\|x^k - x^\star\|^2$$

$$\Longrightarrow$$

$$\|x^k - x^\star\|^2 \leq \frac{1}{2^k}\|x^0 - x^\star\|^2$$

$$\mathbf{E}\left[\frac{\|x^{k+1} - x^\star\|^2}{\|x^k - x^\star\|^2}\right] \leq \left(1 - \min\left(\frac{1}{3}, A\right)\right)^k \frac{\Phi^0}{\mu^2} \cdot c$$

$$\mathbf{E}[\Phi^k] \leq \left(1 - \min\left(\frac{1}{3}, A\right)\right)^k \Phi^0$$

Federated Newton Learn (2022) (Existing)

Can we practically use the FedNL implementation presented at ICML 2022?

Federated Newton Learn (2022) (Existing)

Can we practically use the FedNL implementation presented at ICML 2022?

Not Yet!

Requires 4.8 hours to launch a single experiment on a server-grade workstation

The prototype supports only a multi-node simulation

Prototype integration into resource-constrained applications is challenging

Problem is Deeper

Fact #1 from Data-intensive Computing and OS

Overheads introduced from a **large** number of components lead to degradation

[1] Scalability! but at what COST?" by McSherry et.al., **2015**

Fact #2 from Computer Architecture/Programming Languages

Scripting languages offer the advantage of democratizing implementations.

But their **eco-system clashes too much with the principles of the real hardware.**

[2] There's plenty of room at the top: What will drive computer performance after Moore's law? by Leiserson et al., Science ***2020***

Fact #3 from S. Boyd: *This is awesome! I want to use whatever you so-called Control. What should I use? …**nothing**.*

[3] InControl podcast: Interview with Stephen Boyd, ***2023****, 01:17:10*

Fact #4 from P. Liang: *We are in a crisis. Researchers are disconnected from underlying technologies through **abstractions**. The problem is **abstractions are leaky**.*

[4] P. Liang, Stanford CS336 Language Modeling from Scratch, Spring ***2025***

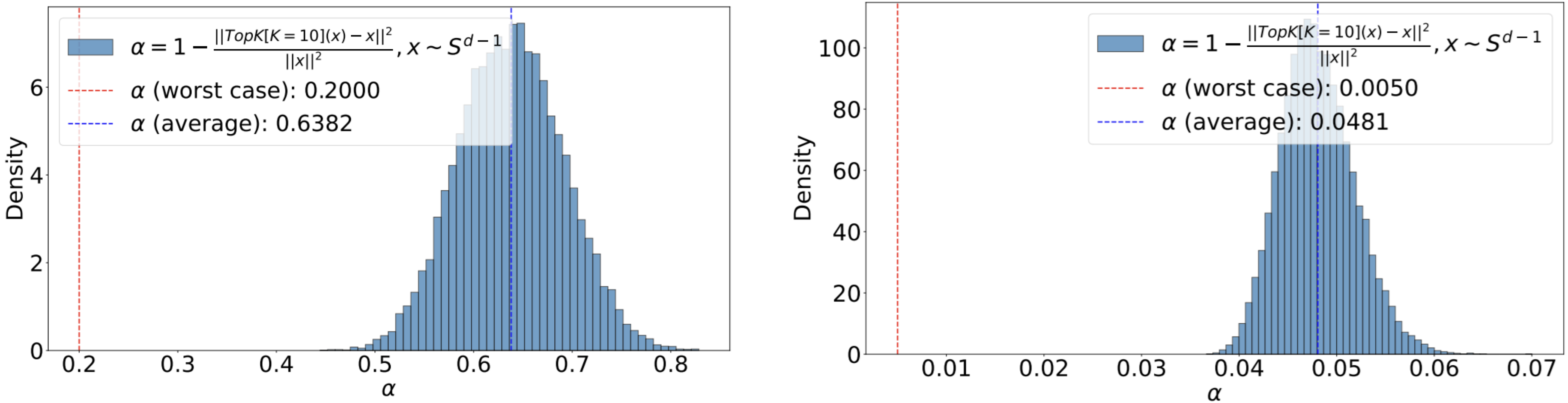
Simultaneously advancing a **rigorous theoretical framework** and an **efficient implementation** presents a significant challenge, as both are equally demanding

Contributions for Making FedNL Practical

1. We proposed two new practical compressors
2. Reduced wall clock time of a baseline by **×1000**
3. Outperforms several best practice solutions
4. Complete independence on 3rd party frameworks
   [Linux, Win, macOS] x [AArch64, x86-64, CUDA]
5. Can be utilized as native OS executable binaries and libraries

Pessimism of TopK Contraction Factor

$$\mathcal{C}^d(\alpha) = \{C \,|\, C : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|C(x) - x\|^2\right] \le (1-\alpha)\|x\|^2\}, \quad \forall \alpha \in (0, 1]$$



(a) d=50, TopK with $k = 10$. (b) d=2000, TopK with $k = 10$.

Figure 7.1: Discrepancy between worst-case α and $\alpha(x)$ when $x \sim_{\text{u.a.r.}} S^{d-1}$. Number of trials 20 000.

Adaptive TopLEK Compressor

$$\mathbb{E}\left[\|TopLEK(x) - x\|^2\right] = (1-\alpha)\|x\|^2, \quad 0 < \alpha \leq 1$$

- The idea is to perform compression using **TopK**, with smaller parameter $\widehat{k} \leq K$ compressing as much as theoretically allowed — but no more
- For **TopLEK**, the inequality that describes contraction becomes a tight equality

Cache-aware RanSeqK Compressor

$$\mathcal{B}^d(\omega) = \{B \mid B : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|B(x) - x\|^2\right] \leq \omega\|x\|^2, \quad \mathbb{E}\left[B(x)\right] = x\}, \quad \forall \omega \geq 0$$

RandK selects a subset of coordinates of cardinality k u.a.r. from a total of d coordinates, zeroing out the rest and scaling the output to preserve unbiasedness

Cache-aware RanSeqK Compressor

$$\mathcal{B}^d(\omega) = \{B \,|\, B : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbb{E}\left[\|B(x) - x\|^2\right] \le \omega\|x\|^2, \quad \mathbb{E}\left[B(x)\right] = x\}, \quad \forall \omega \ge 0$$

RanSeqK selects a pivot s u.a.r. from $\{1,2,\ldots,d\}$, then selects deterministically a block of size $\boldsymbol{k}$ from $(1,2,\ldots,d)$ seen as a torus

It has the same variance as **RandK**, but it is more appealing in practice

Table 7.8: Memory latency comparison in computing devices.

Device and Memory Level	**Approximate Latency (ns)**	**Scale**
CPU cycle	0.3	$x1$
CPU register (SRAM)	0.3	$\times 1$
L1 cache (SRAM)	0.9	$\times 3$
Floating Point addition, subtraction, and multiplication	1.2	$\times 4$
L2 cache (SRAM)	3	$\times 10$
L3 cache (SRAM)	10	$\times 33$
Main memory or Physical Memory (DRAM)	100	$\times 330$
The OS System Call: Transitioning from user to kernel space	300	$\times 1000$
Solid-State Disk (SSD)	10 000	$\times 33\,000$
Rotational Hard Disk Drive (HDD)	10 000 000	$\times 33\,000\,000$

Single Node Experiments: L2 Regularized Logistic Regression

Baseline Improvements

Table 7.1: Single-node setting, $n = 142$, FedNL (B), $r = 1000$, $\lambda = 0.001$, α - option 2, FP64, 24 cores at 3.3 GHz.

#	Client Compression	$\|\nabla f(x^{last})\|$	Total Time (seconds)
1	RandK[K=8d] (We)	$3 \cdot 10^{-18}$	18.84
2	RandK[$k = 8d$] (Base)	$3 \cdot 10^{-18}$	17 510.00
3	TopK[K=8d] (We)	$2.80 \cdot 10^{-18}$	18.72
4	TopK[$k = 8d$] (Base)	$2.80 \cdot 10^{-18}$	19 770.00
5	RandSeqK[K=8d] (We)	$3.19 \cdot 10^{-18}$	16.70
6	TopLEK[K=8d] (We)	$3.45 \cdot 10^{-18}$	18.48
7	Natural (We)	$3.10 \cdot 10^{-18}$	27.02
8	Ident (We)	$2.46 \cdot 10^{-18}$	24.12

Single Node

Table 7.2: Single-node setting, $n = 142$, FedNL-LS (B), $\|\nabla f(x^{last})\| \approx 9 \cdot 10^{-10}$, FP64, 24 cores at 3.3 GHz.

#	Solver	W8A, $d = 301$, $n_i = 350$	A9A, $d = 124$, $n_i = 229$	PHISHING, $d = 69$, $n_i = 77$
	Initialization Time (seconds)			
1	CVXPY	+2.54	+2.33	+2.28
2	FedNL	+0.939	+0.196	+0.081
	Solving Time (seconds)			
3	CLARABEL	19.24	10.83	2.50
4	ECOS	22.22	8.02	2.55
5	ECOS-BB	22.00	8.00	2.12
6	SCS	31.14	19.36	4.57
7	MOSEK	16.90	9.59	3.55
8	FedNL-LS / RandK[$k = 8d$]	4.35	0.34	0.12
9	FedNL-LS / RandSeqK[$k = 8d$]	3.34s	0.29	0.06
10	FedNL-LS / TopK[$k = 8d$]	4.49	0.46	0.10
11	FedNL-LS / TopLEK[$k = 8d$]	4.79	0.34	0.61
12	FedNL-LS / Natural	3.13	0.17	0.08
13	FedNL-LS / Identical	0.63	0.09	0.06

Multi Node Experiments: L2 Regularized Logistic Regression

Table 7.3: Multi-node setting, $n = 50$ clients, 1 master, $|\nabla f(x^{last})| \approx 10^{-9}$, FP64, 1 CPU core/node.

#	Solution	W8A $d = 301$, $n_i = 994$	A9A $d = 124$, $n_i = 651$	PHISHING $d = 69$, $n_i = 221$
Initialization Time (seconds)				
1	Ray	+52.0		
2	Apache Spark	+25.82		
3	FedNL	+1.1		
Solving Time (seconds)				
4	Ray	116.17	28.13	11.54
5	Apache Spark	36.65	33.59	33.14
6	FedNL / RandK[$k = 8d$]	12.6	4.52	0.21
7	FedNL / RandSeqK[$k = 8d$]	12.56	5.10	0.14
8	FedNL / TopK[$k = 8d$]	12.20	5.79	5.23
9	FedNL / TopLEK[$k = 8d$]	15.11	3.26	0.82
10	FedNL / Natural	5.75	1.56	0.14

Ch7: Structure of x1000 Time Improvement for Single Node
FP64, 3.3Ghz, 12 cores, Intel(R) CPU. Logistic Regression d=301

Baseline: Single node Python/NumPy implementation from the original paper	x1
Rewrite in pure C++20/CMake with support macOS, Linux, Windows	x20
Data Processing Optimization	x1.077
Eliminating Integer Division	x1.225
Utilizing AVX512 CPU Extension in x86-64	x1.379
Compiler and Linker Optimization	x1.128
Use Sparsity from FedNL Compressors	x1.44
Reuse Computation from Oracles	x1.50
Basic Linear Algebra Improvements	x1.338
Linear System Solve Improvement	x1.31
Custom oracles without using Cache-Oblivious matrix multiplication	x3.072
Better Compressors Implementation	x1.14
Thread pool of workers equal to number of physical cores, atomics for sync	x1.412
Mem. Optimization. Custom client-specific memory pool instead of global	x1.278

Total number of improvements at a finer granularity: **62**

Federated Learning Challenges Addressed in the Thesis

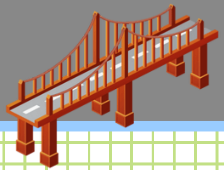
Theoretical Work | **Theory-Inspired Practical Work** | **Practical Work**

1. Data Heterogeneity
2. Device Heterogeneity
3. Communication Bottleneck
4. Privacy
5. Software

Ch1: Introduction

Ch2: FL_PyTorch
Burlachenko et al., 2021

Ch3: EF21-W
Richtárik et al., 2024



Ch3: EF21-W
Richtárik et al., 2024

Ch4: DCGD/PERMK/AES
Burlachenko et al., 2023

Ch5: PAGE Extensions
Tyurin et al., 2023

Ch6: Compressed L2GD
Bergou et al., 2023

Ch7: Unlocking FedNL
Burlachenko and Richtárik, 2024

…king FedNL
…Richtárik, 2024

Ch8: BurTorch
Burlachenko & Richtárik, 2025

Ch8: BurTorch
Burlachenko & Richtárik, 2025

Ch9: Concluding Remarks: Summary and Future Research

Federated Learning Challenges Addressed in the Thesis

Theoretical Work

Theory-Inspired Practical Work

Practical Work

1. Data Heterogeneity
2. Device Heterogeneity
3. Communication Bottleneck
4. Privacy
5. Software

Ch1: Introduction

Ch2: FL_PyTorch
Burlachenko et al., 2021

Ch3: EF21-W
Richtárik et al., 2024

Ch3: EF21-W
Richtárik et al., 2024

Ch4: DCGD/PERMK/AES
Burlachenko et al., 2023

Ch5: PAGE Extensions
Tyurin et al., 2023

Ch6: Compressed L2GD
Bergou et al., 2023

Ch6: Compressed L2GD
Bergou et al., 2023

Ch7: Unlocking FedNL
Burlachenko and Richtárik, 2024

Ch7: Unlocking FedNL
Burlachenko & Richtárik, 2024

Ch8: BurTorch
Burlachenko & Richtárik, 2025

Ch8: BurTorch
Burlachenko & Richtárik, 2025

Ch9: Concluding Remarks: Summary and Future Research

Chapters Excluded from In-Depth Presentation

Ch2: FL_Pytorch (2021) Existing software frameworks for FL prioritize deployment, raise the entry barrier, and demand expertise in distributed systems. Research requires tools with functionalities distinct from industrial runtimes.

Ch5: PAGE Extensions (2023) PAGE is a theoretical optimal algorithm for finding a stationary point in sampled gradient complexity in big - $\mathcal{O}$ notation. This work enhances the analysis of PAGE and extends it with other sampling strategies.

Ch6: Compressed L2GD (2023) New Paradigm for FL was proposed in *"Federated Learning of a Mixture of Global and Local Models"* (2021) by F.Hanzely and P.Richtárik. This work extends it with bidirectional communication compressors.

Ch8: BurTorch (2025) Latency-efficient backpropagation CPU implementation, which outperforms: JAX, TF, TF Lite, LibTorch (C++), PyTorch TorchScript, PyTorch Python, Apple MLX, Autograd, Micrograd in memory, time, consumed energy.

Thank You for Your Time and Attention!

All presented projects are accompanied by open-source code, promoting a culture of openness and collaboration

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

open source
initiative®

Backup Slides

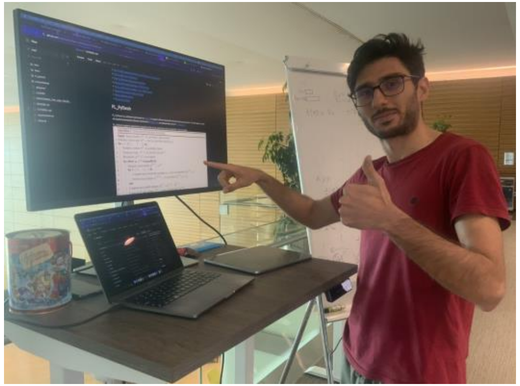
Ch2: FL_Pytorch

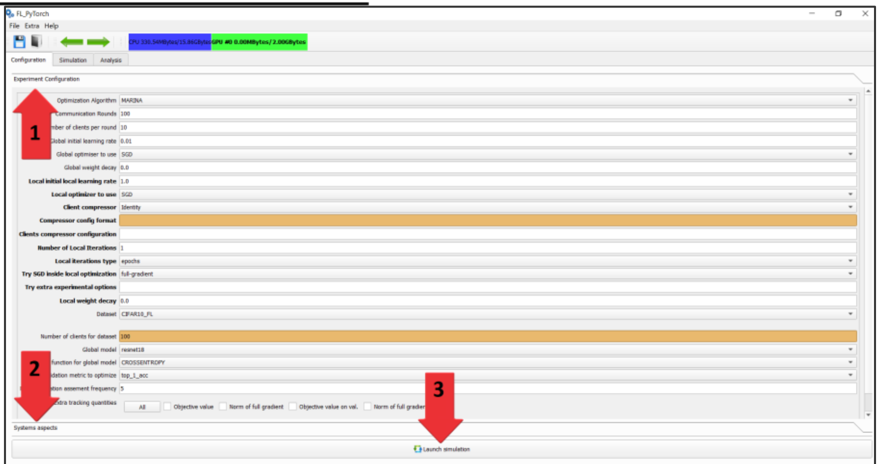
TOOL	Docs.	DX	Lang.	AI frameworks	AI type	Examples	Dist. channels	Multinode?	C/V	H/V	Sync/Async	PVC/SEC	Tools integration	TOTAL
Flower [49]	1.5	1.5	2	2.5	2	1	1	1	1	0	1	1	1.5	17
OpenFL [50]	1.5	1.5	2	2	1	1	1	1	1	1	1	1	1.5	16.5
IBM-Federated [45]	1.5	1	1	2	2	1	1	1	1	0	0	1	1.5	14
PySyft [46]	1.5	1	1	2	1	1	1	1	1.5	0	0	1	1.5	13.5
Nvidia Flare [43]	1	1	1.5	2	2	1	1	1	1	0	0	1	1	13.5
FedML [48]	1.5	0	1	2.5	2	1	1	1	1	0	1	0	1.5	13.5
Fedn [60]	1	1	2	2	1	1	0	1	0.5	1	0	1	1.5	13
Fedlearn-algo [54]	0	0	1.5	2	2	1	0	1	1	1	0	1	1.5	[illegible]
XFL [81]	1	0	1	2	2	1	0	1	1	1	0	1	1	12
PLATO [80]	1	0	1	2.5	1	1	1	1	1	0	1	0	1.5	12
FATE [47]	1	0	1.5	0	2	1	0	1	1	1	1	1	1.5	12
APPFL [62]	1	0	2	1	1	1	1	1	1	0	1	0	1	11
FedLab [51]	1	0	2	1	1	1	1	1	1	0	1	1	0	11
FedBioMed [61] (GitLab)	0	1	1	2	1	1	0	1	0.5	0	0	1	1.5	10
FedJAX [55]	1	1.5	1	2	1	1	1	0.5	0	0	1	0	0	10
OpenFED [37]	1	1	2	1	1	1	1	0.5	0	0	0	1	0	9.5
Tensorflow Federated [44]	1	1	2	1	1	1	1	0.5	0	0	1	0	0	9.5
PyVertical [56] [57]	0	1	2	0	1	1	0	1	1	0	0	1	1	9
FL-Pytorch [71]	1.5	1	1	1	1	1	1	0.5	0	0	0	0	0	8
FLUTE [79]	0	0	1	1	1	1	0	0.5	0	0	1	0	1	7.5
PriMIA [58]	1	0	0	1	1	1	0	0.5	0	0	0	1	1.5	7
Sunday FL [66]	1	0	1.5	0	0	1	0	1	1	0	0	0	1	6.5
dsMTL [65]	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0	1.0	0.0	6
Substra [59]	1	0	0	0	0	1	1	1	1	0	1	0	0	6
DecFL [67]	0	0	0	1	1	1	0	1	0.5	0	0	1	0	5.5
Vantage6 [69]	1.5	0	1	0	0	0	1	1	1	0	0	0	0	5.5
HyFed [63]	0	1	0	0	0	1	0	1	0	0	0	1	0	4
MTC-ETH [68]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.5	0	0	1.0	0.0	2.5

FeLebrities (2023):

A User-Centric Assessment of Federated Learning Frameworks

W. Riviera,
I.B. Galazzo,
G. Menegaz





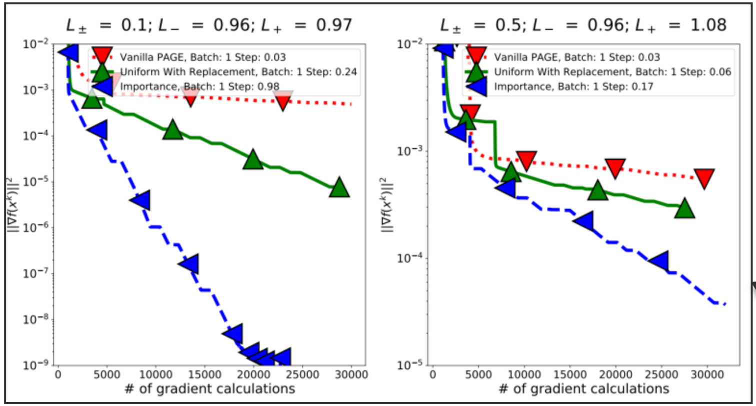
Key Step in Discovery Path

The Ω is the probability sample space. Let $S : a_1 \times \cdots \times a_n \cdots \Omega \to m$ with the following properties:

1. $\mathbb{E}_w[S(\cdot)] = \frac{1}{n}\sum_{i=1}^n a_i$
2. $\mathbb{E}_w\left[\|S(\cdot) - \frac{1}{n}\sum_{i=1}^n a_i\|^2\right] = \frac{A}{n}\sum_{i=1}^n \left(\frac{1}{nw_i}\|a_i\|^2\right) - B\|\frac{1}{n}\sum_{i=1}^n a_i\|^2$

In the paper, we provide several sampling strategies that satisfy the above conditions. Next, with respect to the weighting w_i, we require to estimate the constants $L_{+,w}$ and $L_{\pm,w}$:

1. $\frac{1}{n}\sum_{i=1}^n \frac{1}{nw_i}\|\nabla f_i(x) - \nabla f_i(y)\|^2 \le L_{+,w}^2\|x-y\|^2$
2. $\frac{1}{n}\sum_{i=1}^n \frac{1}{nw_i}\left(\|\nabla f_i(x) - \nabla f_i(y)\|^2 - \|\nabla f(x) - \nabla f(y)\|^2\right) \le L_{\pm,w}^2\|x-y\|^2$



Comparison on synthetized quadratics

Theoretical Improvements

Our Theory for Single Gradient Oracles:

$$N = \mathcal{O}\left(n + \frac{\Delta_0}{\epsilon^2}\cdot|S|\left(L_- + \sqrt{\frac{n}{|S|}((A-B)L_{+,w}^2 + BL_{\pm,w}^2)}\right)\right)$$

Original PAGE Analysis:

$$N_{\text{orig}} = \mathcal{O}\left(n + \frac{\Delta_0}{\epsilon^2}\cdot(L_- + \sqrt{n}L_+)\right)$$

Improved Original PAGE Analysis:

$$N_{\text{improved}} = \mathcal{O}\left(n + \frac{\Delta_0}{\epsilon^2}\cdot(0 + \sqrt{n}L_+)\right)$$

Important Sampling in PAGE:

$$N_{\text{important-sampling}} = \mathcal{O}\left(n + \frac{\Delta_0}{\epsilon^2}\cdot\tau\cdot\left(L_- + \sqrt{\frac{n}{\tau}}L_{\pm,w}\right)\right)$$

$$L_{\pm,w} \le L_+ \quad \text{and} \quad L_{+,w} \le L_+$$

Ch6: Compressed L2GD

$$\min_{x_1,\dots,x_n\in\mathbb{R}^d}\left\{F_\lambda(x) := \left(\frac{1}{n}\sum_{i=1}^n f_i(x_i;D_i)\right) + \lambda\cdot\frac{1}{2n}\sum_{i=1}^n \|x_i-\bar{x}\|^2\right\}$$

$f(x) := \frac{1}{n}\sum_{i=1}^n f_i(x_i;D_i)$ $h(x) := \lambda\cdot\frac{1}{2n}\sum_{i=1}^n \|x_i-\bar{x}\|^2$

Goals

- Strongly convex case $\mathbb{E}\left[x^k-x^\star\right]\le\varepsilon\|x^0-x^\star\|^2$
- Non-convex case $\left(\mathbb{E}\left[\|\nabla F(x^k)\|\right]\right)^2\le\mathbb{E}\left[\|\nabla F(x^k)\|^2\right]\le\varepsilon^2$

Compressors

Each client and master use its own unbiased compressor from $\mathcal{B}^d(w)$

Aspects of L2GD

- λ is scalar parameter which allows tradeoff between local and global model
  $\lambda\to 0$ ($\lambda\to+\infty$) all nodes are working in decoupled (coupled) form
- p is probability of making (relaxed) aggregation
- $1-p$ is a probability to make in parallel local GD steps

Algorithm 15 Compressed L2GD.

Input: step size $\eta>0$, probability p
Initialize: $\{x_i^0\}_{i=1,\dots,n}$, $\xi_{-1}=1$, $\bar{x}^{-1}=\frac{1}{n}\sum_{i=1}^n x_i^0$
for $k=0,1,2,\dots$ **do**
  Draw: $\xi_k=1$ with probability p
  if $\xi_k=0$ **then**
    on all devices: $x_i^{k+1}=x_i^k-\frac{\eta}{n(1-p)}\nabla f_i(x_i^k)$ for $i\in[n]$
  else
    if $\xi_{k-1}=0$ **then**
      on all devices: Compress x_i^k to $\mathcal{C}_i(x_i^k)$ and communicate $\mathcal{C}_i(x_i^k)$ to the master
      on master:
        1. Receive $\mathcal{C}_i(x_i^k)$ from all devices $i\in[n]$
        2. Compute $\bar{y}^k\overset{\text{def}}{=}\frac{1}{n}\sum_{j=1}^n\mathcal{C}_j(x_j^k)$
        3. Compress $\bar{y}^k$ to $\mathcal{C}_M(\bar{y}^k)$
        4. Communicate $\mathcal{C}_M(\bar{y}^k)$ to all devices
      on all devices: Perform aggregation step $x_i^{k+1}=x_i^k-\frac{\eta\lambda}{np}(x_i^k-\mathcal{C}_M(\bar{y}^k))$
    else
      on all devices:
        a. $\bar{x}^k=\bar{x}^{k-1}$
        b. Perform aggregation step $x_i^{k+1}=x_i^k-\frac{\eta\lambda}{np}(x_i^k-\bar{x}^k)$
    end if
  end if
end for

$\eta\cdot\frac{1}{1-p}\cdot\nabla f(x)$, w.p. $1-p$

$\approx\eta\cdot\frac{1}{p}\cdot\lambda\cdot\nabla h(x)$, w.p. p

Results

1. The work extends paper [HR, 2021] with bidirectional unbiased compressors
2. Linear convergence rate to neighborhood (strongly convex case) and theory for non-convex case
3. Optimal values of $p(\lambda, L := nL_f)$
4. Extended empirical study
5. Highlighted that **FedAVG** is a particular case of **L2GD** when $\eta\lambda\approx np$

Table 6.2: Summary of the benchmarks. The measured quantity is bits/n to achieve 0.7 Top1 test accuracy, with $n=10$ clients. For DenseNet-121, MobileNet, ResNet-18 the baseline is FedAVG with Natural compressor with 1 local epoch.

Model	Training Parameters	L2GD bits/n	Baseline bits/n
DenseNet-121	79×10^5	8×10^{11}	$4\cdot10^{15}$
MobileNet	32×10^5	1.7×10^{11}	1×10^{15}
ResNet-18	11×10^6	1.1×10^{12}	1.5×10^{16}

Table 8.2: Backpropagation over 100K iterations with a tiny compute graph from Figure 8.1. Mean and std. deviation across 5 launches, FP64, Windows OS. See also Figure 8.3. The numerical results across frameworks match exactly.

#	Framework, Mode, Language	Device	Compute Time (sec.)	Relative to BurTorch
1	BurTorch, Eager, C++	CPU	0.007 ± 0.0004	×1.0 (We)
2	TensorFlow 2.8.0, Eager, Python	CPU	55.217 ± 0.2975	×7 888.1
3	TensorFlow 2.8.0, Graph, Semi-Python	CPU	14.469 ± 0.0734	×2 067.0
4	TF Lite 2.8.0, Graph, TF Lite Interpreter	CPU	0.589 ± 0.0102	×84
5	Autograd 1.7.0, Eager, Python	CPU	18.956 ± 0.2962	×2 708.0
6	PyTorch 2.5.1, Eager, Python	**GPU**	51.380 ± 0.4666	×7 340.0
7	PyTorch 2.5.1, Eager, Python	CPU	10.419 ± 0.0647	×1 488.4
8	PyTorch 2.5.1, Graph, TorchScript	CPU	9.994 ± 0.1021	×1 428.5
9	PyTorch 2.5.1, Eager, LibTorch, C++	CPU	5.300 ± 0.0667	×757.14
10	JAX 0.4.30, Eager, Python	CPU	291.764 ± 8.5373	×41 860.5
11	JAX 0.4.30, Graph, Semi-Python	CPU	5.580 ± 0.0661	×797.1
12	Micrograd, Eager, Python	CPU	1.590 ± 0.0152	×227.1
13	In Theory for this CPU (Registers Only)	CPU	$\Omega(0.0004)$	×0.057

Table 8.7: BurTorch and PyTorch in training GPT-3 like model, FP32, 1 CPU core, Peak private virtual memory. Trainable variables: 46K.

Batch	BurTorch, Eager, C++		PyTorch, Graph, TorchScript		PyTorch, Eager, Python	
	Compute (ms)	Mem. (MB)	Compute (ms)	Mem. (MB)	Compute (ms)	Mem. (MB)
1	0.515 ± 0.067	16.7	11.119 ± 48.118	1 624	11.715 ± 10.741	1 300
2	1.027 ± 0.091	16.7	11.177 ± 37.138	1 623	12.166 ± 11.461	1 300
4	2.106 ± 0.130	16.7	11.762 ± 37.171	1 624	12.424 ± 11.120	1 300
8	4.222 ± 0.238	16.7	12.041 ± 36.312	1 631	13.167 ± 11.613	1 308
16	8.358 ± 0.644	16.7	13.451 ± 37.415	1 633	14.111 ± 11.278	1 308
32	16.787 ± 1.03601	16.7	16.048 ± 36.460	1 632	16.661 ± 11.122	1 308
64	31.696 ± 0.737	16.8	21.794 ± 37.302	1 640	22.189 ± 11.531	1 316

Table 8.5: Comparison of BurTorch and PyTorch performance for training MLP-like model. Batch: $b = 1$, Compute: FP32, Single CPU core. Initialization time is end-to-end time for training with 1 iteration. Compute time excludes batch preparation. Memory is the peak private virtual memory.

#	Parameters (d) / Hidden Dim.(e)	PyTorch, Eager, v2.5.1 [CPU]			BurTorch, Eager [CPU]		
		Init (ms)	Compute (ms)	Mem. (MB)	Init (ms)	Compute (ms)	Mem. (MB)
1	5,963 ($e = 4$)	5 540	1.46 ± 4.63	2 651	15.63	0.032 ± 0.008	35.8
2	18,587 ($e = 16$)	5 627	1.52 ± 4.21	2 653	16.51	0.074 ± 0.016	36.7
3	35,419 ($e = 32$)	5 673	1.55 ± 5.00	2 653	18.24	0.124 ± 0.019	38.3
4	69,083 ($e = 64$)	5 537	1.63 ± 4.62	2 668	18.94	0.221 ± 0.040	40.8
5	136,411 ($e = 128$)	5 799	1.79 ± 5.19	2 660	21.39	0.417 ± 0.077	45.9
6	540,379 ($e = 512$)	5 556	3.01 ± 5.57	2 683	37.09	2.093 ± 0.429	71.4
7	1,079,003 ($e = 1024$)	5 544	5.57 ± 6.75	2 719	56.57	4.550 ± 0.847	107.0

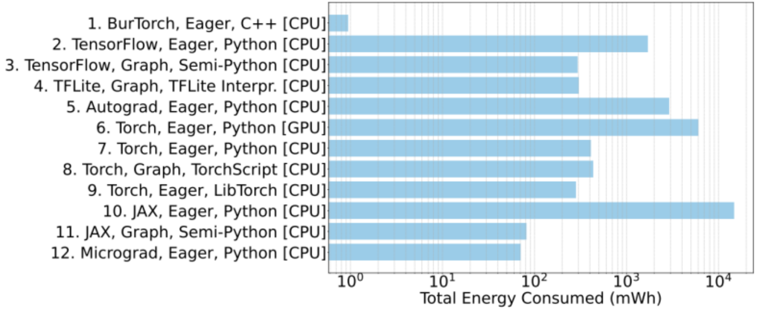
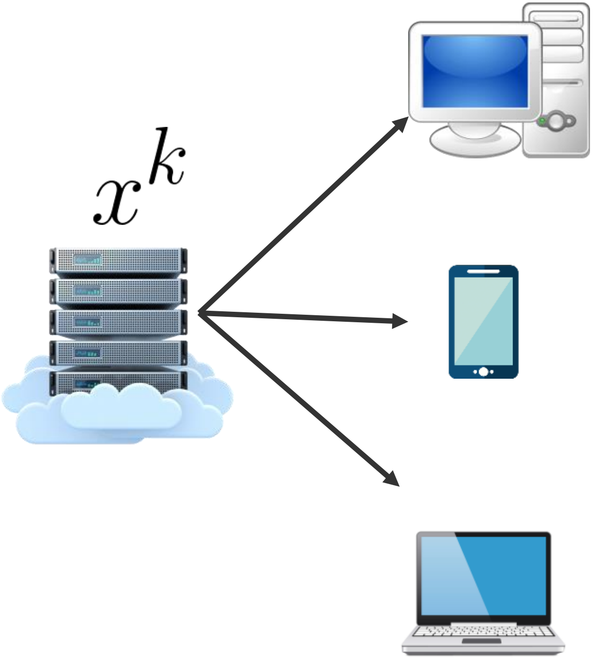


Figure 8.7: Visualization of Table 8.19. Total power drain over 200K iterations with a *small* dynamically constructed compute graph (Figure 8.2) consisting of 32 nodes, using FP64. Voltage: 11.7V, Battery: DELL J8FK941J, Chemistry: Li-poly, OS: Windows 11. The numerical results across frameworks match exactly.



Single-Node	Typical Bandwith
CPU <-> System DDR Memory	51 200 MBytes/sec (DDR5)
GPU core <-> GPU DDR Memory	128 000 MBytes/sec (GDDR6) DRAM in NVIDIA GPU NVIDIA Ada Lovelace 1008 000 MBytes/sec
GPU DDR <-> PCI-E <-> System DDR Memory	4 000 MBytes / sec (PCI-E v5, 1 lane)
SATA 3x (HDD)	6000 Mbytes / sec
USB 3.0 (External storage)	600 MBytes / sec
GPU <-> GPU (NVLink)	50 000 Mbytes/sec
GPU <-> GPU (NVLink via NVSwitch inside DGX-2)	50 000 Mbytes/sec

Multi-Node	Typical Bandwith
Fast Ethernet	12.5 MBytes/sec
Gigabit Ethernet	125 MBytes/sec
InfiniBand HDR	6250 Mbytes/sec
InfiniBand Melanox	25 000 Mbytes/sec

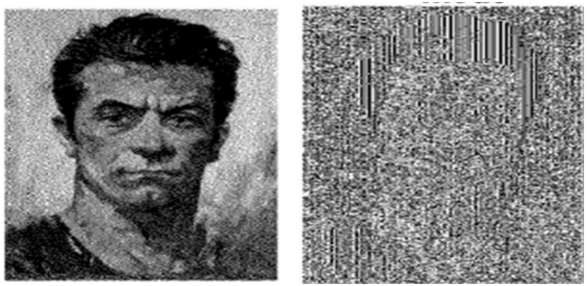
AES is secure for encoding a **single block**
For multiple blocks, it should be used with a
Mode of Operation Algorithm

Overhead: 16 bytes/message

AES with EAX Authentication

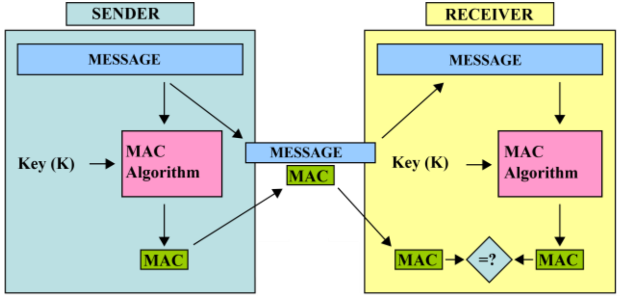
Overhead: 16 bytes/message

To ensure that message has not been altered in transit,
AES should be paired with a
Message Authentication Code Algorithm
(Similar to CRC for non-secure applications)



Incorrect Electronic Code Book (ECB)

Images: Google Search



Input message $m \in \mathbb{R}^{N/2}$

For input with d scalars, amount of ciphertexts is $\lceil \frac{d}{N/2} \rceil$

For AES-128 compatibility level $N > 16\,384$

Encrypt message $m' \in \mathbb{Z}[x]/(x^N + 1)$ into $m'' \in \left(\mathbb{Z}_q[x]/(x^N + 1)\right)^2$

Computation on encrypted messages $(m''_1, m''_2, \dots)$. CKKS allows to perform: addition, multiplication, and rotation.

Ch4: A Day of Life for Message with CKKS Scheme

Input message $m \in \mathbb{R}^{N/2}$

Encode message $m \in \mathbb{R}^{N/2}$ into $m' \in \mathbb{Z}[x]/(x^N + 1)$ ← $2 \cdot \frac{N}{2} \cdot \lceil \frac{d}{N/2} \rceil$ bits

Encrypt message $m' \in \mathbb{Z}[x]/(x^N + 1)$ into $m'' \in \left(\mathbb{Z}_q[x]/(x^N + 1)\right)^2$

Computation on encrypted messages $(m''_1, m''_2, \ldots)$. CKKS allows to perform: addition, multiplication, and rotation.

Ch4: A Day of Life for Message with CKKS Scheme

$\#$ bits to store m'' is: $2 \cdot 2 \cdot \frac{N}{2} \cdot \lceil \frac{d}{N/2} \rceil \cdot Q$

$m''(\text{shared}) = pk(\text{secret}) \cdot m' + \text{noise}$

Encode message $m \in \mathbb{R}^{N/2}$ into $m' \in \mathbb{Z}$

$\#$ bits to store q is $Q = \log_2(q)$

$Q \geq 438$ for AES-128 level

Encrypt message $m' \in \mathbb{Z}[x]/(x^N + 1)$ into $m'' \in \left(\mathbb{Z}_q[x]/(x^N + 1)\right)^2$

Computatio ... otation.

All keys in CKKS are polynomials from $\mathbb{Z}_q[x]/(x^N + 1)$

The key size is $Q \cdot N$ bits

Input message $m \in \mathbb{R}^{N/2}$

↓

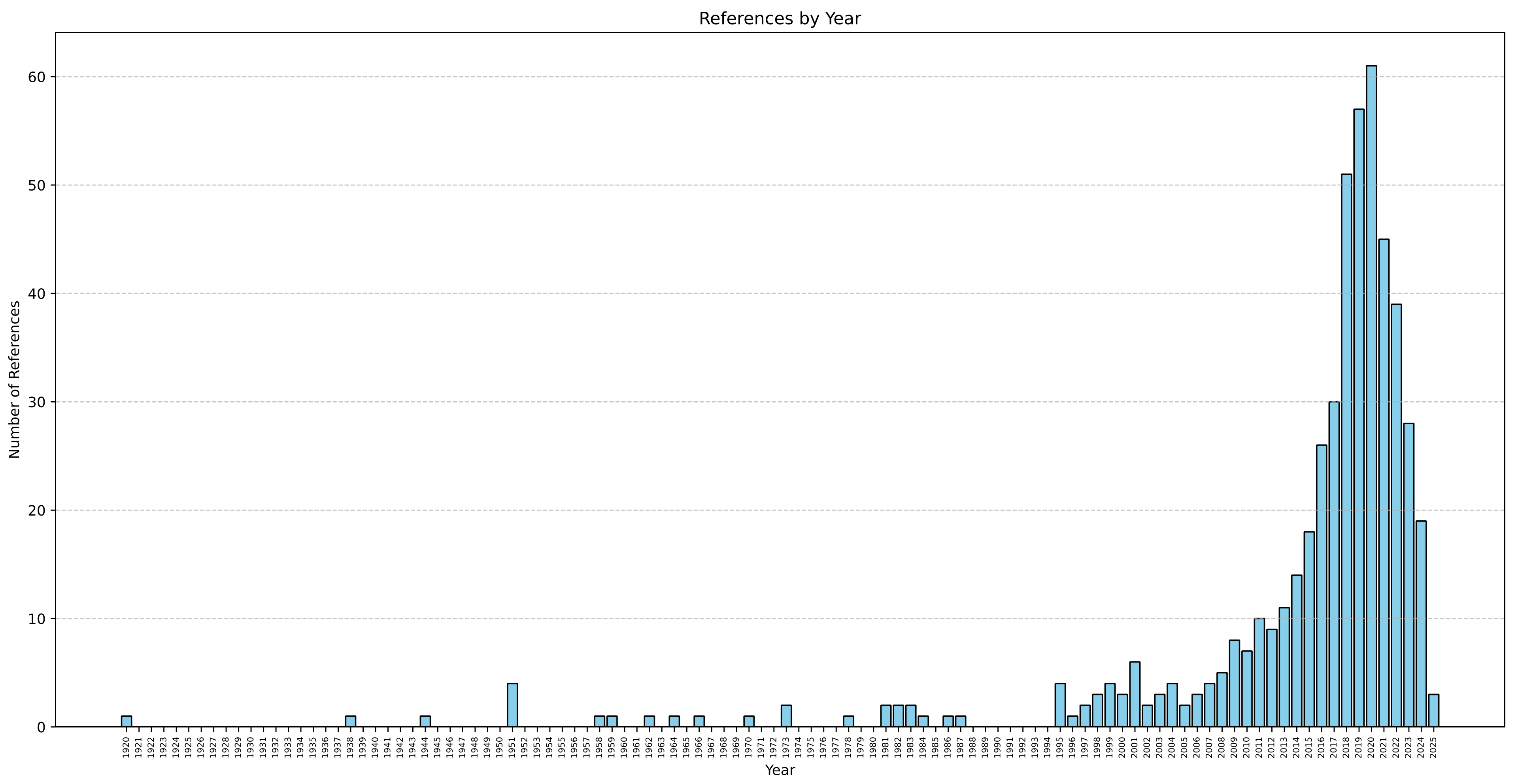
Encode message $m \in \mathbb{R}^{N/2}$ into $m' \in \mathbb{Z}[x]/(x^N + 1)$

↓

Encrypt message $m' \in \mathbb{Z}[x]/(x^N + 1)$ into $m'' \in \left(\mathbb{Z}_q[x]/(x^N + 1)\right)^2$

↓

Computation on encrypted messages $(m''_1, m''_2, \dots)$. CKKS allows to perform: addition, multiplication, and rotation.


References by Year
Number of References
Year
0
10
20
30
40
50
60
1920
2025

Original $n = 4$ clients

$L_1 = 1$

$L_2 = 1$

$L_3 = 1$

$L_4 = 1000$

Original $n = 4$ clients

Example

$L_1 = 1$

$L_2 = 1$

$L_3 = 1$

$L_4 = 1000$

$$L_{\mathrm{AM}} = \frac{1}{4}\left(1 + 1 + 1 + 100\right) = 25.75$$

$$L_{\mathrm{QM}} \approx \sqrt{\frac{1}{4}\left(1 + 1 + 1 + 100 \cdot 100\right)} = 15.73$$

Ch3: EF21 Reloaded (Approach 1)

Original $n = 4$ clients

Cloned $N = 42$ clients

$L_1 = 1$ $\quad \lceil \frac{L_1}{L_{\text{AM}}} \rceil = \lceil \frac{1}{25.75} \rceil = 1$

$L_{11} = 1$

$L_2 = 1$ $\quad \lceil \frac{L_2}{L_{\text{AM}}} \rceil = \lceil \frac{1}{25.75} \rceil = 1$

$L_{21} = 1$

$L_3 = 1$ $\quad \lceil \frac{L_3}{L_{\text{AM}}} \rceil = \lceil \frac{1}{25.75} \rceil = 1$

$L_{31} = 1$

$L_4 = 1000$ $\quad \lceil \frac{L_4}{L_{\text{AM}}} \rceil = \lceil \frac{1000}{25.75} \rceil = 39$

$L_{4j} = 1000/39 = 25.64$

$1 \leq j \leq 39$

Algorithm/Setting	Terminate condition	Iterations #54
Gradient Descent / $f(x)$ is strongly convex. (generalizable via adding regulizers with cheap proximal operator)	$\lvert x^k - x^* \rvert^2 \le \varepsilon \lvert x^0 - x^* \rvert^2$ => if $\nabla f(x^*) = 0$ *then* $\lvert f(x^k) - f(x^*) \rvert \le \frac{L}{2} \lvert x^k - x^* \rvert^2$	$k \ge \frac{L}{\mu} \log\left(\frac{1}{\varepsilon}\right)$
Stochastic Gradient Descent / $f(x)$ is strongly convex / $g(x)$ such that $E[g(x) \mid x] = \nabla f(x)$ / $g(x)$ such that $E[\lvert g(x) - \nabla f(x^*) \rvert^2 \mid x] \le 2AD_f(x, x^*) + C$ $D_f(x, x^*) = f(x) - f(x^*) - < \nabla f(x^*), x - x^* >$	$E[\lvert x^k - x^* \rvert^2] \le \varepsilon \lvert x^0 - x^* \rvert^2$ => if $\nabla f(x^*) = 0$ *then* $\lvert f(x^k) - f(x^*) \rvert \le \frac{L}{2} \lvert x^k - x^* \rvert^2$	$k \ge O\left(\frac{1}{\varepsilon} \cdot \log\left(\frac{1}{\varepsilon}\right)\right)$
Gradient Descent / $f(x)$ is convex.	$f(x^k) - f(x^*) \le \varepsilon \lvert x^0 - x^* \rvert^2$	$k \ge \frac{1}{2\alpha\varepsilon}, \alpha \in \left(0, \frac{1}{L}\right]$
Accelerate Gradient Descent / $f(x)$ is convex.	$f(x^k) - f(x^*) \le \varepsilon \lvert x^0 - x^* \rvert^2$	$k \ge 1 + \sqrt{\frac{2}{\alpha\varepsilon}}, \alpha = \frac{1}{L}$ **(Optimal)**
Stochastic Subgradient Descent / $f(x)$ is convex / $g(x)$ is unbiased / easy prove: g(x) is bounded by G, start iterate x^0 has an upper bound distance R to x^*	$E[f(x^k) - f(x^*)] \le \varepsilon$	$k \ge O\left(\frac{1}{\varepsilon^2}\right)$ with optimal $a_k = \frac{R/G}{\sqrt{k}}$ **(Optimal)**
Stochastic Gradient Descent / $f(x)$ is convex / $g(x)$ is unbiased / g(x) satisfy sigma-k assumption	$E[f(x^k) - f(x^*)] \le \varepsilon$	$k \ge O\left(\frac{1}{\varepsilon}\right)$ to neighborhood. $k \ge O\left(\frac{1}{\varepsilon^2}\right)$ exactly convergence to a solution. **(Optimal)**
Gradient Descent / $f(x)$ is non-convex, but smooth	$\lvert \nabla f(x^k) \rvert \le \varepsilon$	$k \ge O\left(\frac{1}{\varepsilon^2}\right)$
Stochastic Gradient Descent / $f(x)$ is non-convex, but smooth.	$E \lvert \nabla f(x^k) \rvert \le \varepsilon$	From $k \ge O\left(\frac{1}{\varepsilon^2}\right)$ to $k \ge O\left(\frac{1}{\varepsilon^4}\right)$ depending on assumptions
Optimal SGD for case when $f(x)$ is finite sum of n functions / $f(x)$ is non-convex, but smooth (e.g. PAGE) PAGE reduces to GD if p=1 or $\tau = n$	$\lvert \nabla f(x^k) \rvert \le \varepsilon$	$k \ge O\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$ **(Optimal)**